Degree in Industrial Technologies and Smart Industry
MII+MIC

Master's final project

Lane Detection Model Based on Deep Learning Algorithms

Author
Juan Antolín Tello del Rosal

Supervised by
Tilak Singh

Madrid
2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título ...Lane Detection Model Using Deep Learning Algorithms..... en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico ...2020..... es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Juan Antolín Tello Fecha: 26 / 08 / 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Singh, Tilak Fecha: 25 / 08 / 2020

Degree in Industrial Technologies and Smart Industry
MII+MIC

Master's final project

Lane Detection Model Based on Deep Learning Algorithms

Author
Juan Antolín Tello del Rosal

Supervised by
Tilak Singh

Madrid
2020

Modelo de Predicción de Carretera mediante Redes Neuronales Profundas

Autor: Tello del Rosal, Juan Antolín

Supervisor: Singh, Tilak

Entidad Colaboradora: Mitsubishi FUSO Bus & Truck Corporation

Resumen del Proyecto

Introducción

En los últimos años el vehiculo autonomo se ha convertido en una realidad. Ahora es fácil ver las diferentes aplicaciones de estos vehiculos autonomos en diferentes escenarios. Desde robots automatizados en un almacen, pasando por drones de reparto hasta vehiculos completamente autonomos capaces de circular sin intervención humana por la carretera.

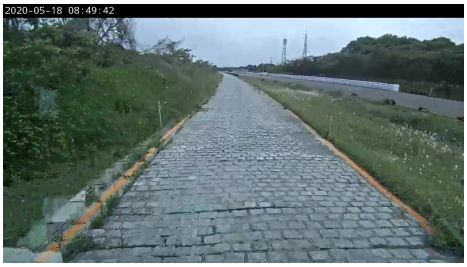
Uno de los aspectos más importantes que existen dentro del ecosistema del vehículo autónomo son los sistemas de detección de objetos durante la conducción, en especial la detección de la calzada.

El principal objetivo de este proyecto es desarrollar un modelo de Detección de vía capaz de detectar la carretera en cualquier situación. Para el desarrollo del modelo se utilizarán Deep Neural Networks, las cuales se basaran en TensorFlow y Keras, y se utilizará Python como lenguage de programación.

El modelo está desarrollado para detectar la carretera en tiempo real. La entrada de video será recibida por una cámara ubicada en la parte frontal del vehículo. El vehículo es un camión FUSO Canter, ubicado en las instalaciones de Kitsuregawa Testing Truck. Estos camiones en los cuales se implantará el modelo final, se utilizan en pruebas de resistencia y durabilidad. El circuito consta de dos carreteras principales rectas con algunas curvas ligeras a lo largo del circuito y dos giros en U o curvas cerradas al final de la carretera, creando un circuito simple para probar los vehículos, pero con gran cantidad de partes diferentes.

La distribución del tipo de vía del circuito es desigual, siendo el Rough Road la parte extensa del mismo. Ambas curvas cerradas son Smooth Road, mientras que solo la mitad de un camino recto está hecho de Smooth Road.

(a) Calzada Asfaltada



(b) Calzada Adoquinada

Figure 1: Diferencia entre Calzada Asfaltada y Calzada Adoquinada

Además de los diferentes tipos de carreteras, el modelo deberá tener un buen rendimiento en los diferentes climatologias e iluminación.

Existen varios métodos de detección de carriles que se han utilizado para desarrollar diferentes modelos durante los últimos 20 años. Los métodos tradicionales con respecto al tema LaneDetection incluían técnicas de detección de bordes y ajuste de líneas para recrear las líneas de la carretera con ecuaciones polinómicas y luego usarlas como predicción.

Con el desarrollo y la introducción de algoritmos de Deep Learning, la investigación sobre detección de carreteras ha alcanzado otro nivel. Los nuevos modelos basados en algoritmos de Deep Learning superan a los métodos tradicionales y son capaces de ofrecer una representación y detección más amplias de la escena de la carretera. De estos modelos de aprendizaje profundo, vale la pena mencionar los modelos convolucionales completos de extremo a extremo, que se centran en el tema de la segmentación semántica y crean una salida de mapa de píxeles de todos los diferentes objetos de la escena. Algunos modelos de segmentación semántica notables son la arquitectura SegNet y la arquitectura ERFNet, ambas con buenos resultados en la segmentación de la imagen de la carretera.

Otros modelos de deep learning, además de las redes convolucionales, incluyen Redes Neuronales Recurrentes para mejorar el proceso de predicción, no solo un frame a la vez, sino también teniendo en cuenta las predicciones anteriores. Las arquitecturas RNN más utilizadas son las LSTM que ayudan a mantener información valiosa de predicciones anteriores.

Metodología

La principal solución para el proyecto Lane Detection fue desarrollar un modelo binario de segmentación semántica, que generará un mapa de segmentación de píxeles de la carretera a partir de una imagen de entrada.

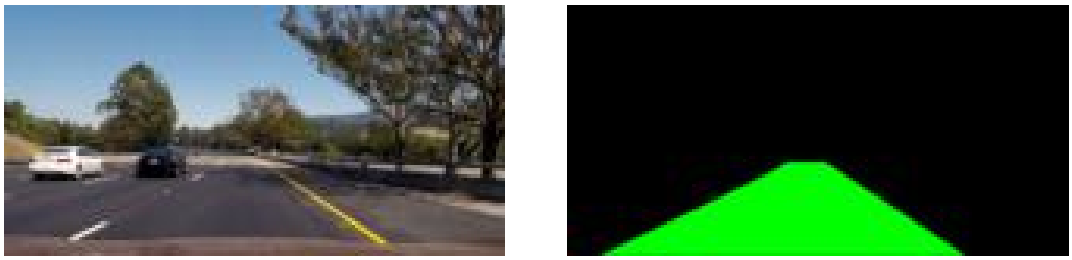
El desarrollo del proyecto se divide en dos fases diferentes. La primera fase es la recopilación de datos y el procesamiento previo de las imagenes y la segunda fase es el modelado y la creación de los diferentes modelos. Estas dos fases son parte de un proceso iterativo, durante la fase de Modelado se requirieron nuevos datos y nuevamente se re-

quirió el Recopilación y Preprocesamiento de Datos antes de continuar con el desarrollo de los diferentes modelos.

Durante la fase de recopilación de datos, el objetivo principal era crear conjuntos de datos adecuados para entrenar los diferentes modelos y preprocesarlos para hacer que las imágenes sin procesar y las etiquetas fueran adecuadas para ser utilizadas como datos de entrenamiento y prueba.

Para el desarrollo del proyecto se utilizaron tres datasets principales y la combinación de ellos.

El primer conjunto de datos fue el conjunto de datos de línea de base, creado por Michael Virgo, un conjunto de datos de Whole Road etiquetado. El conjunto de datos ya estaba comprimido y listo para usarse como imágenes de entrenamiento y etiquetas. El conjunto de datos contó con 12628 imágenes y las etiquetas respectivas. Las etiquetas eran una representación de mapa de píxeles de toda la carretera, con solo un canal de color en lugar de los tres que tienen las imágenes normales. Esta reducción de canal ayudó a disminuir la necesidad computacional del proceso de formación. La resolución de las imágenes y las etiquetas es de 160 por 80 píxeles.

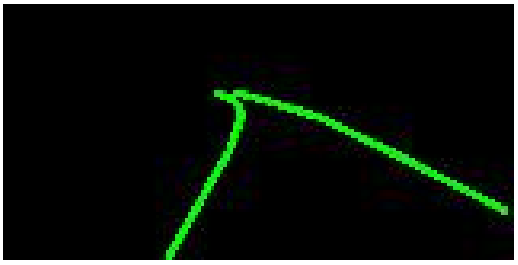


(a) Imagen (b) Etiqueta

Figure 2: Extracto del Baseline Dataset. imagen y su respective Ground Truth

El segundo conjunto de datos utilizado fue el conjunto de datos TuSimple. Este conjunto de datos, comúnmente utilizado en modelos y desafíos de detección de carriles. El conjunto de datos de TuSimple es un conjunto de datos de código abierto propiedad de TuSimple y consta de 3626 fotogramas de video etiquetados. Las etiquetas están contenidas en un archivo JSON. Para crear un mapa semántico binario de píxeles, se procesó el archivo JSON para tomar los datos de las diferentes líneas y, tomando solo los dos carriles del ego, crear un conjunto de datos comprimido de la misma manera que el conjunto de datos de la línea de base para estar listo para implementar como datos de entrenamiento. Después de un poco de procesamiento de imágenes, el número final de imágenes de la base de datos TuSimple son 14504 imágenes y sus respectivas etiquetas.

(a) Frame

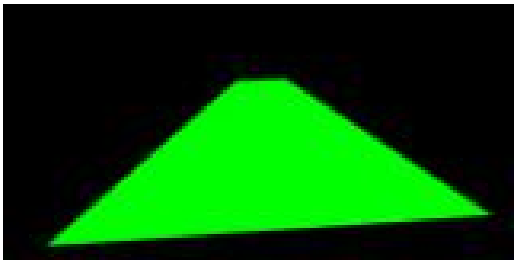


(b) Label

Figure 3: Extracto del TuSimple Dataset. imagen y su respective Ground Truth

El último conjunto de datos utilizado es el conjunto de datos FUSO. Este conjunto de datos se utiliza para reentrenar el modelo y aumentar el conjunto de datos de la línea de base con el metraje del circuito de prueba FUSO para aumentar el rendimiento de los modelos en los escenarios de Rough Road. El conjunto de datos se creó utilizando una herramienta de etiquetado creada exclusivamente para generar el conjunto de datos. El conjunto de datos final de FUSO consta de 504 imágenes y etiquetas del circuito FUSO en escenarios de día despejado y noche despejada. La resolución de imagen es de 160 por 80 píxeles tanto para la etiqueta como para la imagen.

(a) Frame



(b) Label

Figure 4: Extracto del FUSO Dataset. imagen y su respective Ground Truth

Después de la cración de lso diferentes datasets, la siguiente fase del desarrollo es el modelado y la creación de los diferentes modelos.

El primer modelo probado se basó en la arquitectura SegNet. Esta primera primera arquitectura propuesta se basa en una estructura de Encode-Decoder simétrica formada por capas convolucionales 2D con ReLU como función de activación, tamaño de núcleo de 3 por 3 y un tamaño de entrada de imagen de (160,80,3), max pooling y upsampling layers. El Baseline model se entrenó solo con el Baseline dataset. Para el entrenamiento, la función de pérdida utilizada fue el error cuadrático medio, utilizando el optimizador de Adam para calcular los parametros del modelo.

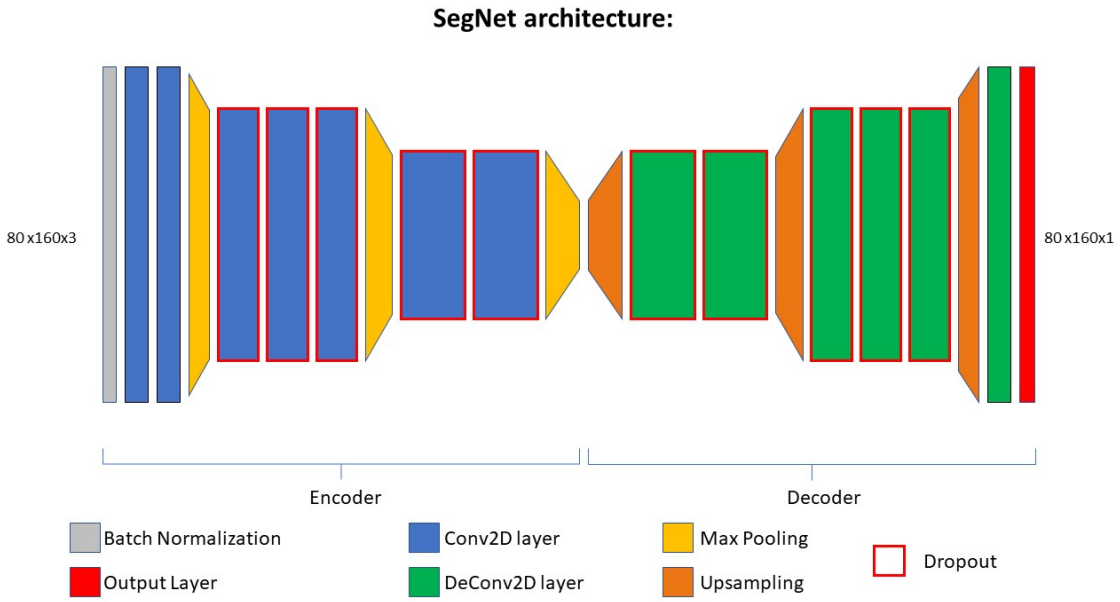


Figure 5: Baseline model architecture

Los resultados obtenidos al probar este modelo no fueron los esperados, más notablemente en escenarios de Noche y en carretera Adoquinada.

(a) Imagen (b) Salida + Imagen

Figure 6: Predicción del Baseline model en escenario de Rough Road

(a) Imagen (b) Salida + Imagen

Figure 7: Predicción del Baseline model en escenario de Noche

Otro modelo fue creado usando la arquitectura SegNet, el modelo TuSimple Base. los resultados obtenidos fueron similares a los del modelo de línea de base. El modelo no fue capaz de detectar correctamente los EgoLanes en ninguna situación y tuvo un rendimiento terrible en los escenarios Rough Road y Night.

Después de los malos resultados obtenidos, el siguiente paso fue ajustar el modelo de línea de base utilizando el conjunto de datos FUSO. Se probaron varios experimentos, cambiando los diferentes parámetros durante el Tune Fitting, y se creó el mejor modelo reentrenado utilizando todo el conjunto de datos FUSO, con una pequeña tasa de aprendizaje de e^{-5} para Adam.

Los resultados son mejores que el modelo de línea de base, a pesar de que el modelo reentrenado tiene algunos problemas en los escenarios nocturnos que predicen como una carretera partes del campo que rodea la carretera, además de la falta de consistencia del mapa de segmentación de píxeles.

(a) Input Image (b) Merged Output

Figure 8: Night footage of the Rough road Retrained prediction

Después de reentrenar el modelo SegNet Baseline con el conjunto de datos FUSO, quedó claro el impacto de los nuevos datos FUSO en el rendimiento del modelo en situaciones de carreteras difíciles.

El siguiente paso fue entrenar una nueva arquitectura, basada en la arquitectura ERFNet pero en lugar de los módulos Resnet, tiene capas convolucionales 2D estándar. Este nuevo modelo se entrenó con la combinación del conjunto de datos de referencia y el conjunto de datos FUSO.

El rendimiento del modelo fue mejor que el de los modelos anteriores y se pudo realizar de manera precisa durante todos los escenarios diferentes, incluso en la carretera Rain Nightin Rough, la más desafiante.

(a) Input (b) Merged Output

Figure 9: ERFNet_reduced Model Output for Rain Night Rough Road

Se probó otro modelo de TuSimple con esta nueva arquitectura. El rendimiento del modelo fue mejor que el SegNet TuSimple. Al igual que los modelos Whole Road, el ERFNet TuSimple fue capaz de dar un resultado preciso en escenarios suaves, pero cuando trató de dar una predicción en escenarios de caminos irregulares, el modelo generó varias líneas, no solo los margenes de la carretera deseados.

(a) Input (b) Merged Output

Figure 10: ERFNet_TuSimple Model Output for Smooth Road

El ERFNet model es el modelo final que será implementado en el Testing Truck en Kitsuregawa.

Resultados

para medir los diferentes resultados y predicciones de los modelos se han utilizado dos métodos.

El primer método consiste en un analisis visual , comparando las diferentes predicciones de los modelos en las imagenes del Test Dataset. En esta comparación es posible observar que el ERFNet model es con diferencia el que mejores resultados obtiene. Mientras que los modelos basados en SegNet no son capaces de aportar un mapa de pixeles constante y denso, el modelo ERFNet siempre genera una prediccion constante y precisa, sin vacios en el mapa de píxeles y las predicciones siempre siguiendo losn margenes del circuito.

Los modelos basados en TuSimple dataset no consiguen buenos resultados en escenarios de Noche y carretera Adoquinada. Por esto, no serán tomados en consideración para ser implementados como modelo final.

El segundo método para comparar el desempeño de los modelos fue un método cuantitativo, para cada modelo se calcularon dos métricas principales. El primero fue el F1, que fue una ponderación tanto del recuerdo como de la precisión del modelo, y el segundo fue el Índice de Jaccard o Intersection over Union (IoU). La intersección $(A \cap B)$ está compuesta por los píxeles que se encuentran tanto en la Ground Truth como enn la predicción, mientras que la unión $(A \cup B)$ está compuesta simplemente por todos los píxeles que se encuentran en la predicción o en la Ground truth.

Las métricas se calcularon comparando la salida de los diferentes modelos con las etiquetas de Ground Truth del Test Dataset. Los resultados obtenidos de los diferentes modelos se pueden observar en la siguiente tabla:

ERFNet TuSImple Evaluation Results

Model	Precision	Recall	IoU	F1
Baseline Model	0.8359	0.8054	0.7021	0.8146
Retrained_5 Model	0.8227	0.9357	0.7798	0.8709
ERFNet_reduced Model	**0.9026**	**0.9603**	**0.8703**	**0.9283**
TuSimple Base Model	0.4705	0.2361	0.1875	0.3071
ERFNet_TuSimple Model	0.5137	0.5030	0.3528	0.5015

Table 1: ERFNet TuSimple model test metrics

Después de ambos métodos utilizados para comparar los modeles, se ha llegado a la conclusión que el ERFNet Model es el que mejor resultados obtiene.

Conclusiones

Durante el desarrollo del proyecto se tomaron en consideración varios aspectos para determinar el mejor modelo posible.

El bajo rendimiento de la arquitectura SegNet puede deberse a dos factores principales. La primera y más fácil de apreciar es la falta de datos de Rough Road en el Baseline Dataset, lo que dificulta que el modelo clasifique la carretera correctamente. El segundo es que la estructura de la red tenía en el centro de la Red una capa de Max Pooling seguida de una capa de Upsampling. Es posible que en esa reducción dimensional de las salidas de las capas convolucionales 2D internas faltara alguna información valiosa de la carretera.

Con la arquitectura ERFNet, este diseño se cambió y fue posible apreciar grandes mejoras en la precisión y consistencia de los resultados. Además, agregar más convoluciones internas con más kernels ayudó a recopilar más información de los diferentes detalles de la imagen.

Otro aspecto importante fue que los valores para las métricas obtenidas entre los modelos de EgoLanes y los modelos de Whole Road eran relativamente grandes, haciendo imposible la comparación entre los modelos basados en esas métricas.

En conclusión, el modelo Final desplegado en el Camión Autónomo de Pruebas es capaz de detectar la Carretera del Circuito de Pruebas en todos los diferentes escenarios propuestos al inicio del proyecto, enfrentando algunos pequeños problemas en los escenarios Nocturnos y algunos problemas intermitentes provocados por la no estabilidad del cámara de entrada.

Algunos posibles mejoras futuras con respecto al modelo real y la detección de carriles son mejorar el rendimiento de salida en giros bruscos, aumentar el número de clases para detectar objetos diferentes además de la carretera, introducir una red LSTM para mejorar la predicción continua y evitar predicciones erraticas entre fotogramas y generar nuevas imagenes con los margenes como Ground truth para mejorar los modelos de EgoLanes.

Road Lane Detection Model using Deep Learning Algorithms

Summary Report

Introduction

Autonomous driving has become a reality during the last years. Nowadays it is easy to see the application of autonomous driving in many different environments. From from automatic robots in a warehouse, through delivery drones, to fully autonomous vehicles capable of moving freely on public roads without the need for any action by the driver.

One of the most important aspects of the autonomous driving ecosystem is the road detection algorithms, especially the Lane detection. The road detection includes all possible objects that will take part during the driving. From all these objects, one of the most crucial parts to detect properly is the road itself.

The main objective of this project is to develop a Lane Detection model capable to detect the road in extreme conditions. Deep Neural Networks will be used to develop the model. To develop these Neural Networks, it was decided to use the TensorFlow workframe, through the Keras library, and deploy all the different required codes in Python.

The model is developed to detect the road in real time. The video input will be received by a camera located in the vehicle. The vehicle is a FUSO Canter truck, located in the Kitsuregawa Testing Truck facility. The trucks are used in enduring and durability testing. The circuit consist of two main straight roads with some slight curves along the roads and two U-turns or sharp turns at the end of the road, creating a simple circuit to test the vehicles.

The circuit road type distribution is uneven, being the Rough Road the extensive part of it. Both sharp turns are Smooth Road, while only half of one straight road is made of Smooth road.

(a) Smooth Road (b) Rough Road

Figure 11: Difference between the Rough Road and the Smooth Road seen from the Truck's camera

Besides the different type of road the model will need to have a good performance in the different weather and lightning scenarios.

There already exist several Lane Detection methods that have been used to develop different models during the last 20 years. The traditional methods regarding the Lane Detection topic included edge detection techniques and line fitting to recreate the road lines with polynomial equations and then use them as the prediction.

With the development and introduction of Deep learning algorithms the Road Detection research has reached another level. The new models based in Deep Learning algorithms outperform the traditional methods and are able to give a wider representation and detection of the road scene. From these deep learning models it is worth to mention the End-to-end Full Convolutional models, which are focused on the semantic segmentation topic and creates a pixel map output of all the different object in the scene. Some noticeable Semantic Segmentation models are the SegNet architecture and the ERFNet architecture, both with good results in the segmentation of vehicle image.

Other deep learning models, in addition to the convolutional networks, they include Recurrent Neural Networks to improve the prediction process, not only one frame at a time, but also taking into consideration the previous predictions. The most used RNN architectures are the LSTM which helps to keep valuable information from previous predictions.

Methodology

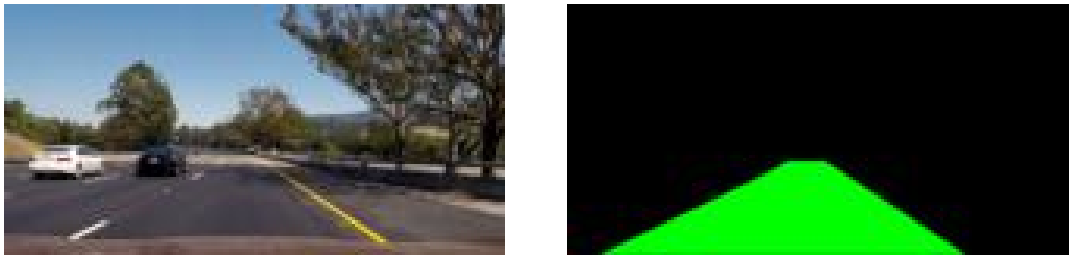
The main solution for the Lane Detection project was to develop a binary Semantic segmentation model, which will generate a pixel segmentation map of the road from an input image.

The development of the project can be divided in two different phases. The first phase is the Data Gathering & Prerpocessing and the second phase is the Modeling. These two phases are part of an iterative process, during the Modeling phase new data was required and again the Data Gathering & Preprocessing was required before to continue with the development of the different models.

During the Data Gathering phase the main objective was to create suitable datasets to train the different models and preprocess them to make the raw images and labels suitable to be used as training and testing data.

For the development of the project three main datasets and the combination of them were used.

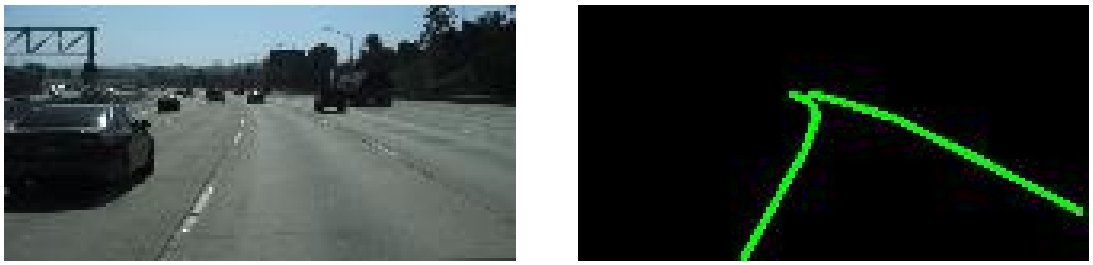
The first dataset was the Baseline dataset, created by Michael Virgo, a Whole Road labeled dataset. The dataset was already compressed and ready to be used as training images and labels. The dataset counted with 12628 images and the respective labels. The labels were a pixel map representation of the whole road, having only one Color channel instead of three as normal images have. This channel reduction helped to decrease the computational need of the training process. The resolution of the images and the labels are 160 by 80 pixels.



(a) Image (b) Label

Figure 12: Extract of the Baseline Dataset. Image and the ground truth label

The second dataset used was the TuSimple dataset. This dataset, commonly used in Lane Detection models and challenges. The TuSimple dataset is an open source dataset owned by TuSimple and consist of 3626 video frames labeld. The labels are contained in a JSON file. To create a semantic binary pixel map, the JSON file was processed to take the data from the different lines and, only taking the two egolanes, create a dataset compressed in the same way as the Baseline dataset to be ready to implement as training data. After some image processing the final number of images of the TuSimple dataset are 14504 images and their respective labels.



(a) Frame (b) Label

Figure 13: Extract of TuSimple dataset. Frame and ground truth label

The last dataset used is the FUSO dataset. This dataset is used to retrain the model and increase the Baseline dataset with the FUSO testing circuit footage to increase the performance of the models in the Rough Road scenarios. The dataset was created using a labeling tool created exclusively to generate the dataset. The final FUSO dataset consists of 504 images and labels of the FUSO circuit in Clear Day and Clear Night scenarios. Image resolution is 160 by 80 pixels for both label and image

The Testing dataset will be also created using this labeling tool. It covers all different scenarios and consist of 240 images and labels, with an image resolution of 160 by 80 pixels.

(a) Frame (b) Label

Figure 14: Extract of FUSO dataset. Frame and ground truth label

After the different datasets were created, the next phase was the Modeling phase.

The first model tried was based on the SegNet architecture. Having as a reference the encoder-decoder structure, the first proposed archotecture was a mirrored Encode-decoder structure formed by 2D Convolutional layers with ReLU as activation function, 3 by 3 kernel size and an image input size of (160,80,3) ,max pooling and upsampling layers. The Baseline model was trained with only with the Baseline Dataset. For the training, the loss function used was the Mean squarred error, using the Adam optimizer to calculate the weights.

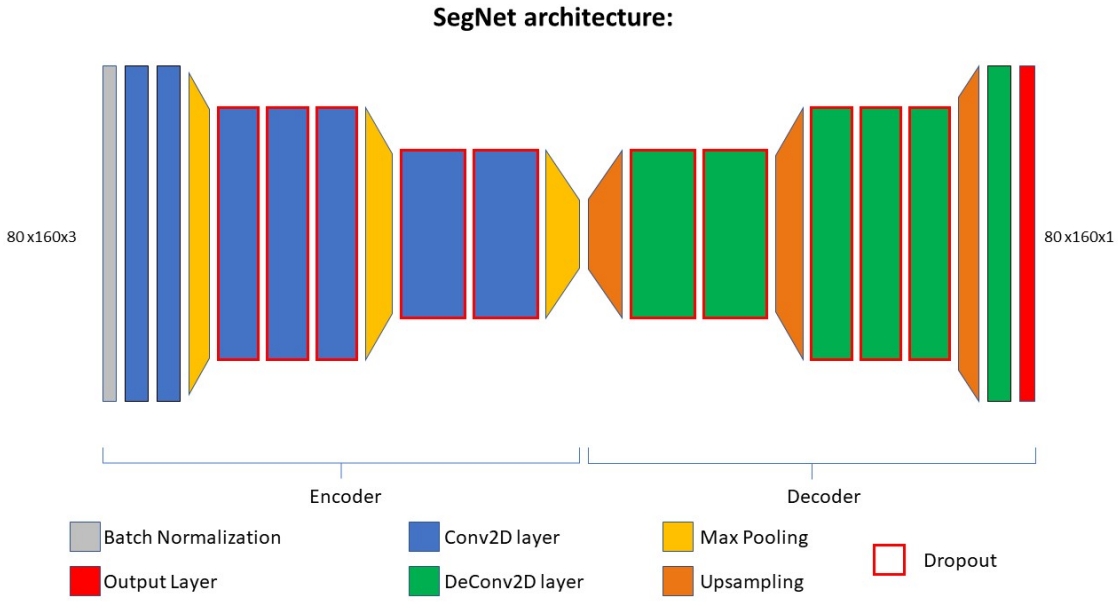


Figure 15: Baseline model architecture

The results obtained with this model were not as good as expected, more noticeable during the Night and Rough Road scenarios:

(a) Input Image (b) Merged Output

Figure 16: Rough Road Baseline model prediction

(a) Input Image (b) Merged Output

Figure 17: Night footage of the Rough road Baseline prediction

Another model was created using the SegNet architecture, the TuSimple Base model. the results obtained were similar as the baseline model. The model was not able to detect properly the EgoLanes in any situation and having an awful output in Rough Road and Night scenarios.

After the bad results obtained, the next step was toTune Fit the Baseline model using the FUSO dataset. Several experiments were tested, changing the different parameters during the Tune Fitting, and the best Retrained model was created using the whole FUSO dataset, with a small learning rate of e^{-5} for Adam.

The results are better the Baseline model, even though the Retrained model has some issues in Night scenarios predicting as a road parts of the field that surrounds the road added to the lack of consistency of the pixel segmentation map.

(a) Input Image (b) Merged Output

Figure 18: Night footage of the Rough road Retrained prediction

After retraining the SegNet Baseline model with the FUSO dataset it was clear the impact of the new FUSO data in the performance of the model in Rough Road situations.

The next step was to training a new architecture, based on the ERFNet architecture but instead of the Resnet Modules, it has standard 2D Convolutional layers. This new model was trained with the combination of the Baseline Dataset and the FUSO dataset.

The performance of the model was better than the previous models traied and was able to perform in an accurate way during all different Scenarios, even in the Rain Night in Rough road, the most challenging one.

(a) Input (b) Merged Output

Figure 19: ERFNet_reduced Model Output for Rain Night Rough Road

Another TuSimple model was tried with this new architecture. The performance of the model was better than the SegNet TuSimple. Same as the Whole Road models, the ERFNet TuSimple was able to give an accurate output in Smooth scenarios, but when it tried to give a prediction in Rough road scenarios, the model generated several lines, not only the desired aEgoLanes.

(a) Input (b) Merged Output

Figure 20: ERFNet_TuSimple Model Output for Smooth Road

The ERFNet model was the final model and was implemented as a road pixel map generator in the Testing Truck in Kitsuregawa.

Results

To measure the accuracy of the different models and determine which was the best model, two main methods were used.

The first method was a qualitative visual method, comparing the output images from the Test Dataset to the labels. It was possible to see that the ERFNet reduced model was by far the best performing model. While the SegNet based models were not able to give a constant and dense segmentation pixel map, the ERFNet model had always a complete pixel map with no holes on it and the predicted outputs were always following the road margins.

The TuSimple models under perform in night situations and in Rough Road scenarios, and were not taken in consideration to be the deployed model.

The second method to compare the models performance was a quantitative method. For each model two main metrics were calculated. The first one was the F1, which was a pondering of both the recall and precision of the model, and the second one was the Jaccard Index or Intersection over Union (IoU). The intersection $(A \cap B)$ is comprised of the pixels found in both the prediction mask and the ground truth mask, whereas the union $(A \cup B)$ is simply comprised of all pixels found in either the prediction or target mask.

The metrics were calculated comparing the Output of the different models with the ground truth labels of the Test dataset.

The results of obtained from the different models can be observed in the following table:

ERFNet TuSImple Evaluation Results

Model	Precision	Recall	IoU	F1
Baseline Model	0.8359	0.8054	0.7021	0.8146
Retrained_5 Model	0.8227	0.9357	0.7798	0.8709
ERFNet_reduced Model	**0.9026**	**0.9603**	**0.8703**	**0.9283**
TuSimple Base Model	0.4705	0.2361	0.1875	0.3071
ERFNet_TuSimple Model	0.5137	0.5030	0.3528	0.5015

Table 2: ERFNet TuSimple model test metrics

After both the qualitative and quantitative testing it is clear that the best performing model is the ERFNet Reduced model.

Conclusions

During the development of the project several aspects were taken in consideration to determine the best possible model.

The low performance of the SegNet archotecture can be due to two main factors. The first one and most easy to apreciate is the lack of Rough road data in the initial dataset, making more difficult for the model to classify the road properly. The second one is the structure of the network had a maxpooling and a upsampling layers together right at the end of the encoder and at the start of the decoder. It was possible that in that dimensional reduction of the outputs of the inner 2DConvolutional layers some valuable information of the road was missing.

With the ERFNet archotecture this layout was changed and it was possible to appreciate high improvements in the accuracy and consistency of the outputs. Also, adding more inner convolutions with more kernels helped to gather more insights from the different details of the image.

Another important aspect was that the values for the metrics obtained between the EgoLanes models and the Whole Road models were relly big, making the comparison between the models based on those metrics impossible.

In conclusion, the Final model deployed at the Autonomous Testing Truck is capableto detect the Road of the Testing Circuit in all different scenarios proposed at the begin-ning of the project, facing some small issues at Night scenarios and some intermittentproblems caused by the non-stability of the input camera

Some possible future steps regarding the actual model and the Lane detection are improve the output performance in sharp turns, Increase the number of classes to detect different object apart from the road, Introduction of LSTM network to improve continuous prediction and avoid missing predictions, and generate new data to improve the EgoLanes models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous driving has become a reality during the last years. Nowadays it is easy to see the application of autonomous driving in many different environments. From from automatic robots in a warehouse, through delivery drones, to fully autonomous vehicles capable of moving freely on public roads without the need for any action by the driver.

The main purpose of the autonomous driving is to create vehicles capable to fulfil different task without human intervention. As final objective, autonomous driving wants to make peoples life easier and more safe.

Lane detection is one of the most important parts of the autonomous vehicle ecosystem. It determines the road that the vehicle must follow and a good lane detecting will provide a huge impact on the autonomous vehicle overall performance. During the last years, with the development of Machine Learning and the improvement of Neural Networks and its architectures, there has been a drastic change on how image processing is being approached. The old standard methods based on computer vision are now being replaced by these new Neural Network based models, which are more powerful and achieve a better results, but having a higher computational power need.

This project will try to implement autonomous driving in a Autonomous Testing Truck, in the FUSO Testing facility located in Kitsuregawa.

"The Kitsuregawa Proving Ground was commissioned in 1980 to provide a world class of test course facilities for validating the development and testing work for trucks and buses. The capabilities of the facilities have been boosted since 1980, and able to perform an ample R & D testing with an addition of anti-lock brake testing circuit, EMI testing wing, engine research wing, and drive train component testing wing. To anticipate and meet future needs and requirements, the facilities are currently engaged in the advance research of engines and other components, vehicles on-road testing, as well as bench testing of functions, performances, durability and reliability." [1]

The main motivation of this project is to avoid human intervention during this testings and avoid putting human lives at risk sometimes, due to the possibility of accidents during this testings and to avoid material loses due to the vehicle accidents.

1.2 Problem Statement

The main objective of this project is to develop a model capable to detect the main road in a robust way, for different road scenarios. This model will be implemented into a private test facility circuit used only by FUSO and owned by FUSO. The project will not be implemented in final release vehicles and will not operate in public environments.

This model will be based in Deep Learning algorithms, specially based in Convolutional Neural Networks architectures. The main objective of the model is to generate a robust semantic segmentation pixel map of the road of the Circuit.

The model will be deployed into Autonomous Testing Truck and will receive the output of the model to determine the deviation of the vehicle in any moment of the road. The model will receive video streaming directly from a camera located inside the vehicle cockpit and will be always focused on the center of the vehicle, giving a direct and central point of view of the road in front of the vehicle.

The vehicle in which the project will be implemented is a FUSO Canter truck, located in the Kitsuregawa Testing Truck facility. The trucks are used in enduring and durability testing. The circuit consist of two main straight roads with some slight curves along the roads and two U-turns or sharp turns at the end of the road, creating a simple circuit to test the vehicles.

The following picture shows an aerial view of the layout of the circuit:

Figure 1.1: Aerial view of the Testing circuit where the model will be implemented

The Testing facility circuit counts with two main types of road:

- Smooth Road: The first type of road, which will be called “Smooth road” from now on, consist in a normal, asphalt road, where can be find in highways and secondary roads all around Japan.

- Rough Road: The second type of road, which will be called “Rough road”, is not made of asphalt. Instead, it is made of paving stone, making the road rougher and not as uniform as the Smooth road. The Rough road also will create some difficulties to the video streaming. Since the road is not made of asphalt, the camera located in the truck will receive all the vibrations the vehicle will experience, making the video footage not steady during this type of road.

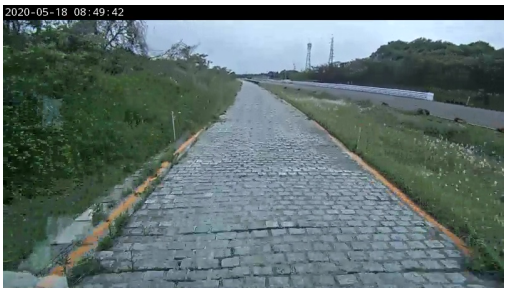
The circuit road type distribution is uneven, being the Rough Road the extensive part of it. Both sharp turns are Smooth Road, while only half of one straight road is made of Smooth road.

The following image shows a representation of the distribution of the road type in the circuit:

Figure 1.2: Road type distribution along the circuit. Red lines shows the Rough Road part of the circuit and the Green lines the Smooth Road.

The following images shows a representation of the Smooth Road and the Rough Road directly extracted from the truck’s camera:

(a) Smooth Road



(b) Rough Road

Figure 1.3: Difference between the Rough Road and the Smooth Road seen from the Truck’s camera

Other challenges the model may face are the not stable recording of the road. Due to the Rough Road irregularities, there are notorious vibrations during the Rough Road situations. The main purpose of these vibrations is to test the performance of the Trucks in extreme conditions. The collateral effect of these vibrations is a non steady recording of the road, having a noticeable variance between frames.

Another challenge faced during the development of the model was the lack of training data available with different road types. All data available to train lane detection models are gathered from highway footage where the roads are all marked by lines and the road is made of asphalt, and from urban areas, with similar characteristics.

1.3 Document structure

The document is structured in 6 different chapters and one appendix. The content of each chapter is explained here:

- **State of the Art**: This chapter explains the different researches done previous the realizatin of the project, giving some information about the Autonoumous Driving current situation, the different Lane Detection methods and a more deep explanation on the Semantic Segmentation models from where the models proposed are based.
- **Datasets Explanation**: This chapter will explain in a detailed way where was the different data used to train and test the different models proposed for the Road Lane Detection project. It includes the different data sources of the footage, the different preprocessing methods used to transform the images into the final datasets and the structure of the different datasets.
- **Models Implementation**: This chapter covers the different approaches tried to develop the models, a detail explanation of the models architectures and some visual representation obtained from the models.
- **Results and Models Comparison**: This chapter includes the resukts and comparison of the different models using two main testing techniques. The first method is a Qualitative method based on the visual representation of the models outputs, comparing the results of the different models according to the desired output, which is get an accurate representation of the road. The second method is a Quantitative method, where using the same testing data for all models, different metrics are calculated with the models predictions.
- **Conclusion and Future Steps**:Final conclusions about the development of the project, the results obtained from the different models and what future steps should be taken to improve the already existing model.

Chapter 2

State of the Art

This chapter gives a description on the main sources and the concepts that have been researched and took as reference during the development of the project.

Lane Detection is one of the most important parts of the Autonomous Vehicles ecosystem and during the last 20 years several approaches have been tried to develop models capable to detect the road in real live situations and with good accuracy.

Before the appearance of the Deep Neural Networks in the Lane detection topic, several types of detection and prediction models existed. These so called traditional Lane Detection methods based the detection of the road by primitive elements such as gradient, color and texture of the image. Two main categories can be distinguish, the Geometrical modeling methods and the Energy minimization methods.

The Geometric modeling methods based their Lane Detection in two steps, first edge detection and then line fitting. Edge detection was achieved using different gradient filters to detect big changes in the color and texture of the input image. Canny Edge Detector [2] model and Gabor Filters [3] are commonly used to generate the edge detection. For line fitting the most used algorithm is Hough Transformation [4]. This algorithm takes the edges generated and transforms them in polynomial functions to be used as the representation of the road. Regarding the Energy minimization models, the most noticeable models were the Kalman Filter and Particle Filter to detect the continuous position of the lanes.

Figure 2.1: Example of the Canny Edge Detection model of an image of a road.

These models kept being the base of the first autonomous vehicles right before starting to implement the new Deep Neural Network models.

2.1 Deep Learning Models

It was with the introduction and the use of Deep Learning algorithms when the Lane Detection topic made a big leap into the real application of Autonomous Vehicles.

A huge variety of deep learning Lane Detection models have been proposed in the past several years. These models can be categorized in three main methods.

2.1.1 Encoder-Decoder Convolutional Neural Network

The encoder-decoder CNN architecture is the main type of network to develop Semantic Segmentation algorithms. These end-to-end architectures are built to detect and separate all elements from the input image in different categories, depending on the desired output of the model.

The main structure of these encoderdecoder architectures are the 2D Convolutional layers. These layers, unlike the regular full connected hidden layers of the regular Neural Networks, do not connect all elements of the image and processes them. These layers take

a threedimensional input, typically an image with three color channels. Then the image is scanned passing a convolution kernel over the image. This kernels works as a filter, inspecting a small window of pixels at a time, for example 3 x 3 in size, and moving the window until they have scanned the entire image. The convolution operation calculates the dot product of the pixel values in the current filter window with the weights defined in the filter.

Using these layers as base, the encoder-decoders take the image as input and the different convolutions gather all the information from all different aspects and features of the input image and use them to create a pixel segmentation map, which will be the classification map of the image.

The main applications of Semantic Segmentation are in autonomous vehicles, human-computer interaction, robotics, and photo editing/creativity tools. There has been created Semantic Segmentation models to focus on specific tasks, like the UNet model [5].

This model was developed to create segmentation of neuronal structures in electron microscopic stacks. Several architectures have been used to create semantic segmentation models to be used in Vehicle road detection.

The main two architectures used as reference in the developing of this model are the SegNet architecture and the ERFNet architecture.

SegNet architecture

The SegNet architecture proposed in 2015 [6], which is a Semantic Segmentation model used to detect the different elements which conform the road scene . This Neural Network architecture is based on a Full Convolutional structure, being an end-to-end full convolutional network, where the Decoder structure is a mirrored version of the Encoder. Is the first architecture chosen as base to develop the Road Detection Model

"The SegNet architecture consists of a sequence of non-linear processing layers (encoders) and a corresponding set of decoders followed by a pixelwise classifier. Typically, each encoder consists of one or more convolutional layers with batch normalisation and a ReLU non-linearity, followed by non-overlapping maxpooling and sub-sampling. The sparse encoding due to the pooling process is upsampled in the decoder using the max-pooling indices in the encoding sequence (see the figure below). One key ingredient of the SegNet is the use of max-pooling indices in the decoders to perform upsampling of low resolution feature maps. This has the important advantages of retaining high frequency details in the segmented images and also reducing the total number of trainable parameters in the decoders. The entire architecture can be trained end-to-end using stochastic gradient descent." [7].

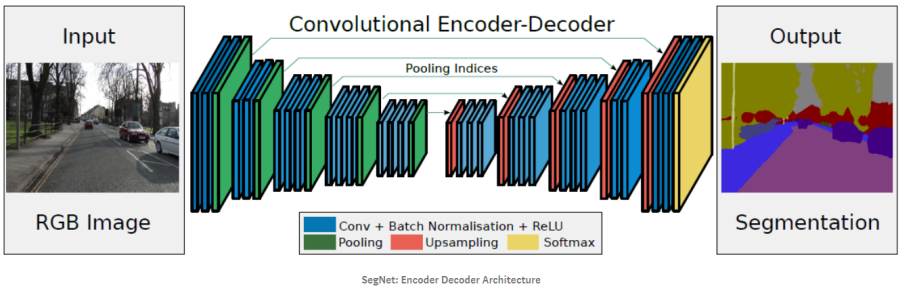


Figure 2.2: SegNet architecture

The SegNet algorithm was trained to be able to predict up to 11 different classes, all forming part of different objects that are part of the road scene and street environment.

The output of the Segnet model is a pixel segmentation map, containing all the elements detected of the road scene and divided in each of the 11 types of object detected.

The next image shows the input and the output of an image fed to the SegNet model:

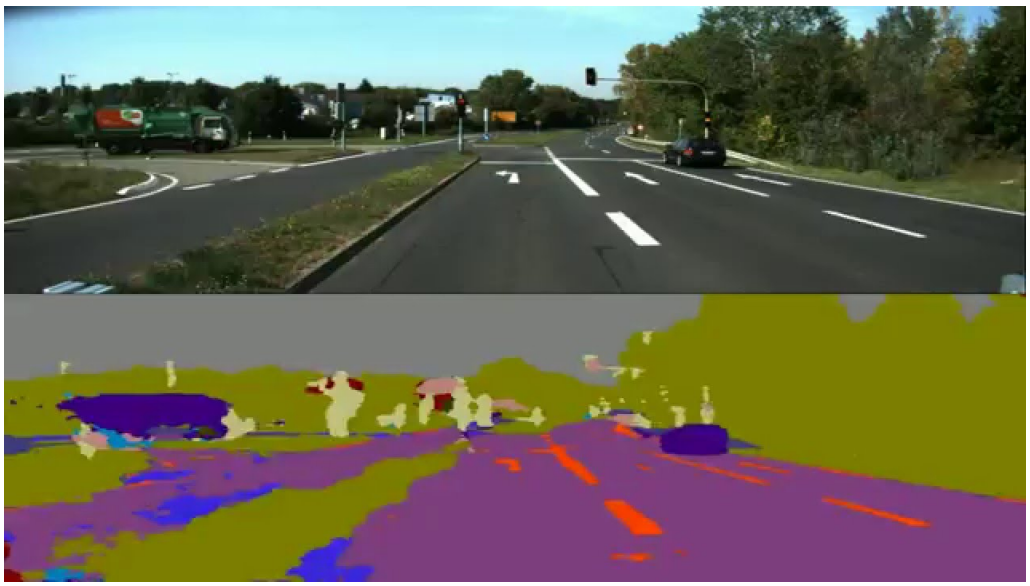


Figure 2.3: SegNet model output. It is possible to observe the semantic segmentation of the different classes the model is able to output

For the development of the Road Lane Detection model only one object will be predicted by the segmentation map, the road, therefore reducing the computational power needed.

ERFNet architecture

The ERFNet model is another Semantic Segmentation approach to be used in vehicle detection, proposed in 2017 [8].

Thestructure of the ERFNet model consisted of a Encoder-Decoder architecture using a specific type of Convolution layers instead of the typical 2D Convolution. These layers are the so called Factorized Resnet Modules with Dilation, which they allow or very deep models to be created without as much risk of vanishing/exploding gradients. Each module include one-dimensional dilated convolutions as base and adding dilated convolutions to give the layers in the network a lot of context. This allowed to have a reduced output time while preserving the input image resolution.

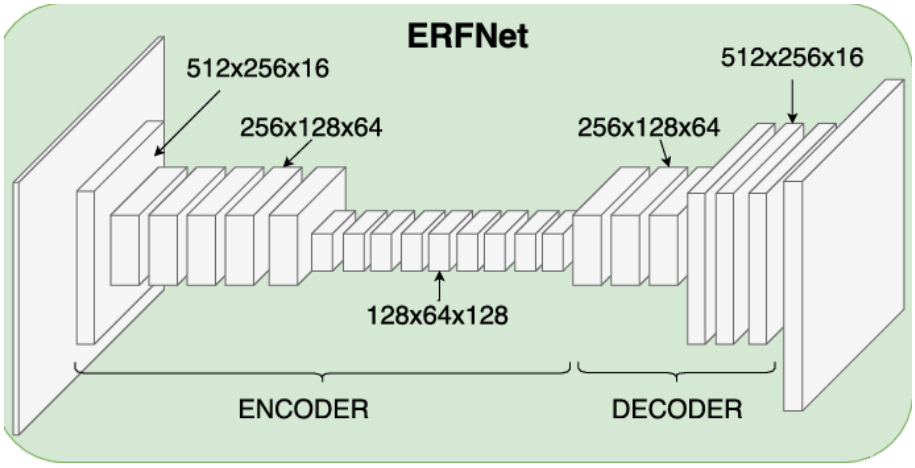


Figure 2.4: ERFNet architecture

This model is able to predict up to 32 different classes of objects.

For the development of the model the model layers were changed. Instead of Factorized Resnet layers, 2D Convolutional Layers were used, and also one object was tried to be detected, the road.

2.1.2 CNN + Recurrent Neural Network

The CNN+RNN models include apart from the Convolutional standard architectures, some recurrent Neural Networks elements to improve the accuracy of the prediction by including the previous predictions as part of the prediction process.

The main reference of CNN+RNN network for future steps that the project is the model proposed Qin Zou et al, in 2018 with their Robust Lane Detection from Continuous Diiving Scenes. [9]

This model architecture consist of three differentiated parts. Apart from the encoder and the decoder, the architecture includes a ConvLSTM block to treat the outputs feature maps from the encoder. In our network, the input and output size of the ConvLSTM are equal to the size of the feature map produced by the encoder. The size of the convolutional kernel is 3 by 3. The ConvLSTM is equipped with 2 hidden layers, and each hidden layer has a dimension of 512. The main objective of the ConvLSTM is to forget the unimportant information for the feature maps extracted from the CNN and remember the essential features from the previous predictions.

The structure of the Robust Lane Detection model is shown in the following image:

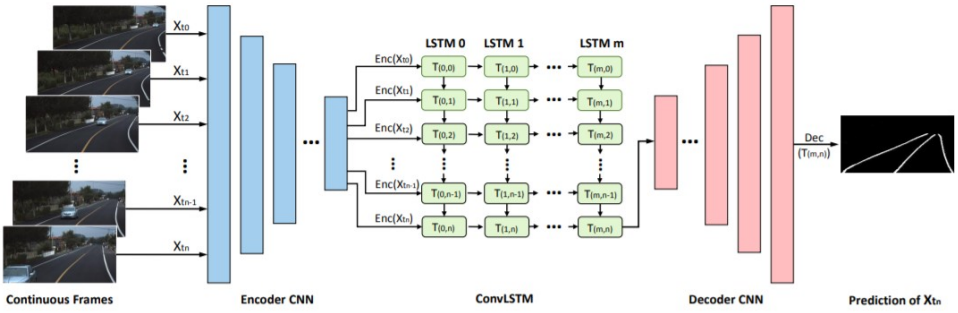


Figure 2.5: Robust Lane Detection architecture

Thanks to the introduction of the LSTM module, the results of the model outperforms other models based on the lane detection with the TuSimple dataset.

Chapter 3

Data Sets Explanation

This chapter will consist in a summary of all footage used to create all the different data-sets, including all different transformation and methods utilized to transform the data-sets used to train the models.

To develop the project 3 different datasets have been used. From these 3 datasets, the two first datasets, Baseline dataset and TuSimple dataset were gathered from the internet. The third dataset, the FUSO dataset, was created from data owned by Daimler Asia FUSO.

In the next sections, a detailed explanation of the different datasets is given.

3.1 Baseline Dataset

The first dataset used to train the model was the Baseline Dataset.

This dataset is obtained from the article posted in Toward Data Science, Lane Detection with Deep Learning [10]. To access the dataset is it possible to download it from the GitHub page where it is allocated [11].

This dataset is property of Michael Virgo and it was used to develop the baseline model

This Dataset (Baseline dataset from now) consist of images extracted from road videos taken with a smartphone attached to the front of a vehicle. The resolution of the initial images taken directly from the video footage was 720p, taken in landscape mode and with a 1280x720 pixels resolution. This image resolution will increase enormously the computational power needed to train the model and after the model is trained, and since the model will be implemented in a real time scenario, all images and labels were downsized to a 160x80 resolution to make all process faster and for the model to be able to mantain a good performance working in real time.

To reduce even more the computational needs and the size of the dataset the channels

of the labels were reduced from 3 (RGB), to just 1, being the final shape of each label (80, 160, 1), and each image (80, 160, 3).

The dataset was initially made out of 2127 images.Each image of the dataset was labeled following the same pattern. For the labels it was decided to create a pixelmap of the whole road, being the label a representation of this road only in the Green channel. Each image label had the whole main egolane road as ground truth, completing this way the dataset with 2127 images and 2127 labels.

Here is an example of one image an the ground truth label. The images are already downsized to 160x80 pixels.

Figure 3.1: Image Example from Baseline Dataset

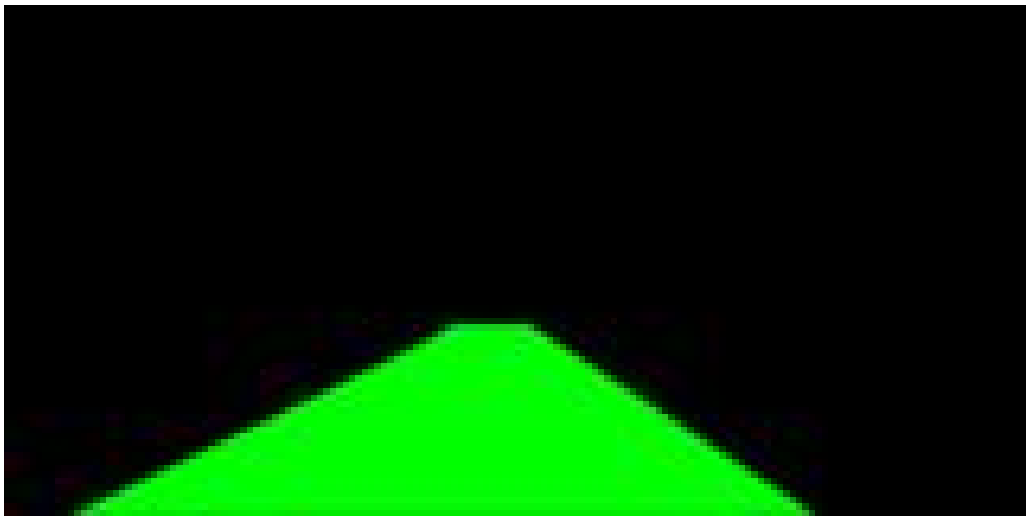


Figure 3.2: Label Example from Baseline Dataset

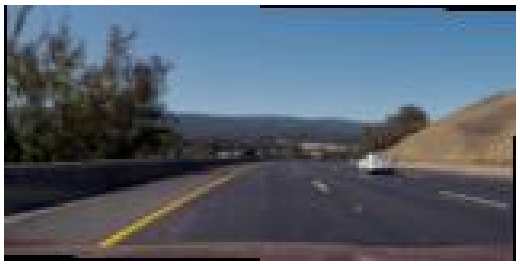
Since the dataset is not very big, image transformation is used to increase the number of images of the dataset.

The image transformation used to transform the main images in the dataset were tilting each image and each label by 3 degrees in both directions, clockwise and anticlockwise. After tilting all images and labels, they were mirrored, duplicating the amount of images of in the dataset.

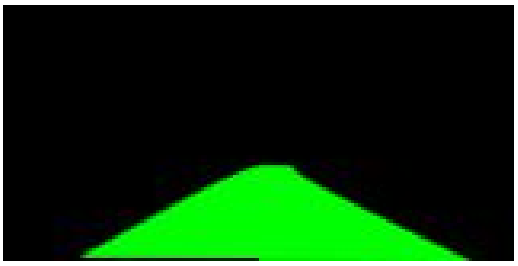
The final dataset consisted in a total of 12762 images for training and testing all different models using this dataset.

Each training dataset, images and labels, were saved in pickle format to save storage and make easier the loading and saving of the data. The images were saved in the file full_CNN_images.p and the labels in the file full_CNN_labels.p .

The final size of each part of the dataset were 479 MB for the image file and 160 MB for the label file. The following images show a series of frames and their labels.



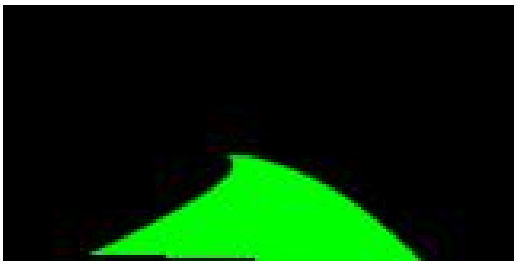
(a) Frame 1



(b) Label 1

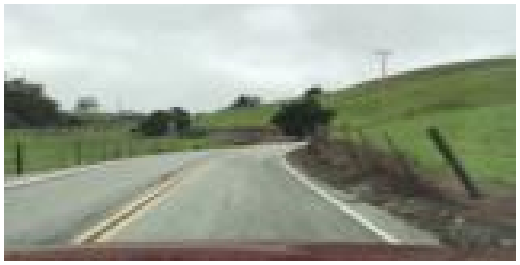
Figure 3.3: Frame and label 1 of Baseline dataset

(a) Frame 1

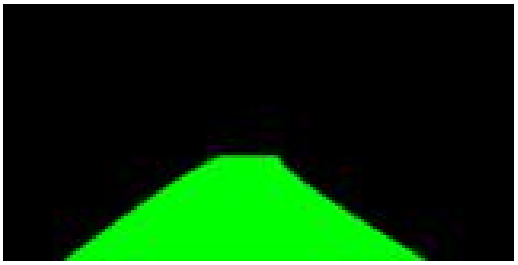


(b) Label 1

Figure 3.4: Frame and label 2 of Baseline dataset



(a) Frame 1



(b) Label 1

Figure 3.5: Frame and label 3 of Baseline dataset

(a) Frame 1 (b) Label 1

Figure 3.6: Frame and label 4 of Baseline dataset

3.2 TuSimple Dataset

The second dataset used on the development of the model is the TuSimple dataset.

The TuSimple Dataset is a dataset owned by the company TuSimple, a self-driving truck company focused on the development of selfdriving heavy-duty trucks. This dataset was released as part of the TuSimple Lane Detection Challenge. It consist of 3626 video clips of 1 sec duration each, with each video clip containing 20 frames. The quality of each frame is 1280 x 720 pixels, in RGB format. From these frames the last frame is labeled, each line labeled on its own. The dataset can be obtained for free as a public download form the Github page of TuSimple [12]. Once the train data is downloaded the directory structure looks like the figure below:

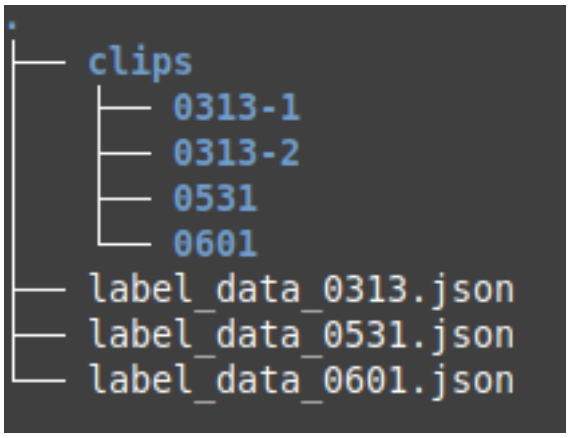


Figure 3.7: TuSimple Dataset directory structure

The dataset is divided in 4 folders, each one subdivided in several ones. Each subdirectory contains the 20 frames of each clip, and the last frame, frame 20th, is the labeled

frame.

The files label_data_(date).json contain the information of all labels inside the main directories. The structure of the json file is the following:

- **raw_file**: string type. the file path in the clip
- **lanes:** list of list. Each list corresponds to a lane, being the first two always the egolanes. In each list, every item consist of the x-coordinate of the lane ground-truth.
- **h_samples:** list of the height values corresponding to the lanes. Each element of the lane is a y-coordinate of the lane ground-truth.

In this dataset, at most four lanes are annotated - the two ego lanes (two lane boundaries in which the vehicle is currently located) and the lanes to the right and left of ego lanes. All the lanes are annotated at an equal interval of height, therefore h_samples contain only one list whose elements correspond to y-coordinates for all lists in lanes. For a point in h_samples, if there is no lane at the location, its corresponding x-coordinate has -2. For example, a line in the JSON file looks like :


```
{"lanes": [[-2, -2, -2, -2, -2, -2, -2, -2, 499, 484, 470, 453, 435, 418, 400, 374, 346, 318, 290, 262, 235, 207, 179, 151, 123,
96, 68, 40, 12, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2],
[-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, 529, 531, 533, 536, 538, 540, 540, 538, 536, 531, 525, 519, 513, 507, 499, 491, 483,
475, 467, 459, 451, 443, 435, 426, 418, 410, 402, 394, 386, 378, 370, 362, 354, 346, 338, 330, 322, 314, 306, 297, 289, 281,
273, 265, 257, 249, 241], [-2, -2, -2, -2, -2, -2, -2, -2, 553, 568, 583, 598, 613, 640, 667, 693, 719, 740, 761, 783, 804,
825, 846, 868, 883, 897, 912, 926, 941, 955, 969, 984, 998, 1013, 1027, 1042, 1056, 1070, 1085, 1099, 1114, 1128, 1143,
1157, 1171, 1186, 1200, 1215, 1229, 1244, 1258, 1272, -2, -2, -2, -2], [-2, -2, -2, -2, -2, -2, -2, -2, 558, 585, 613, 646,
679, 714, 770, 817, 865, 912, 954, 994, 1033, 1073, 1113, 1153, 1193, 1232, 1272, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2,
-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2]],
"h_samples": [160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370,
 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620,
 630, 640, 650, 660, 670, 680, 690, 700, 710],
 "raw_file": "clips/0531/1492626287507231547/20.jpg"}
```


Figure 3.8: TuSimple Dataset json file structure. This represents one label

First TuSimple dataset

To create a dataset from this footage the first attempt was to recreate the labels as an image, creating an np.array matrix which will be the representation of the labels. Each line will be made from several pixels, but since there are only few reference point for each lane the label generated was not good to be used to create a pixel segmentation map of the road images.

The next picture corresponds to a labeled frame from the TuSimple dataset. The picture after it correspond to its label, represented as it is collected from the json file.

In 3.10 it is hard to see the lines, due to their constitution. They are only made from few pixels distributed wide apart from each other. This label is not suitable to be used to train the lane detection model.

Figure 3.9: Frame from TuSimple Dataset

To create a suitable dataset it is necessary to process each label to increment the number of pixels in each label. For this the first solution applied to create the first labeld TuSimple dataset (TuSimple_train_images and TuSimple_train_labels), was to create a pixel cloud around each point of each lane, to augment the number of pixels that were part of the ground truth.

After creating the clusters around each pixel that are part of each lane the result is the following:

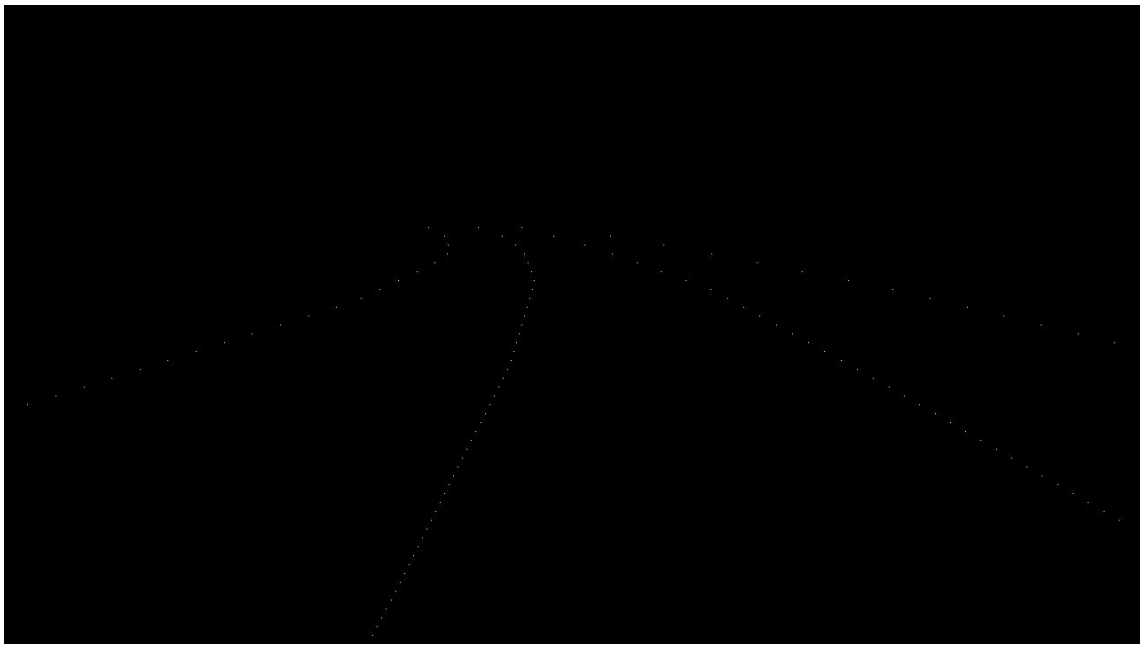


Figure 3.10: Label from previous frame of TuSimple Dataset

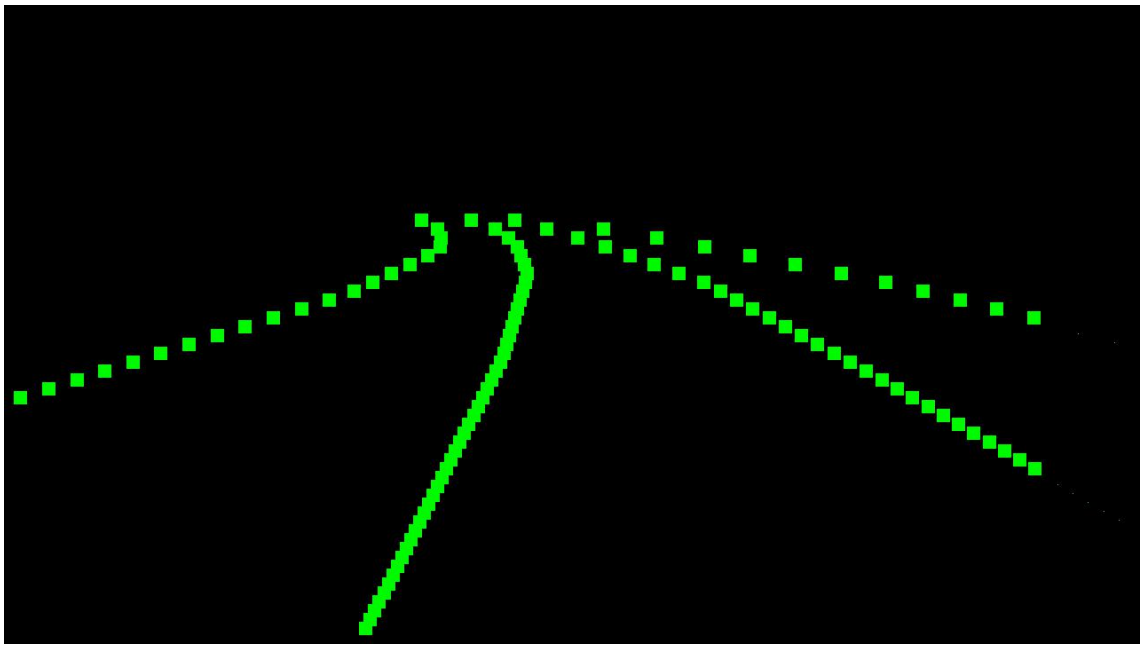


Figure 3.11: Label reshaped of frame of TuSimple Dataset

With this configuration the new label was more suitable to be implemented as the output of the Neural Network.

For the creation of the first dataset, TuSimple_train_ dataset,only the 20th frame of each clip was taken, resulting on a total of 3626 images, all labeled with their ground truth. Since the number of images was not very numerous all images were transformed to increase the size of the training dataset.

The image transformation used to transform the main image in the dataset were mirroring the images horizontally and tilting the image by 4 degrees to each side. After the image transformation the final dataset consisted of 14504 images and labels.

Due to the high quality of the images, the dataset at high quality was too big. Before saving the dataset in a compressed file, each image and each label was resized to 160x80 pixels, to reduce the amount of data needed to storage the dataset. Also, the channels of the labels were reduced from 3 (RGB), to just 1, being the final shape of each label (80, 160, 1), and each image (80, 160, 3).

Each training dataset, images and labels, were saved in pickle format to save storage and make easier the loading and saving of the data. The final size of each dataset were 543 MB for the TuSimple_train_images.p and 181 MB for the TuSimple_train_labels.p .

This training dataset was used to develop the first TuSimple model.

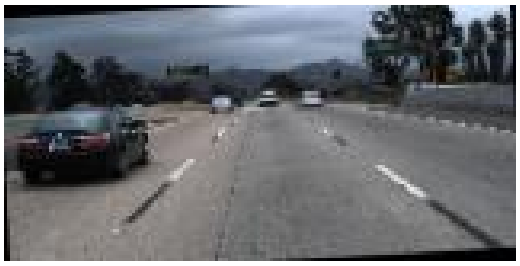
The following images show a series of frames and their labels.

(a) Frame 1 (b) Label 1

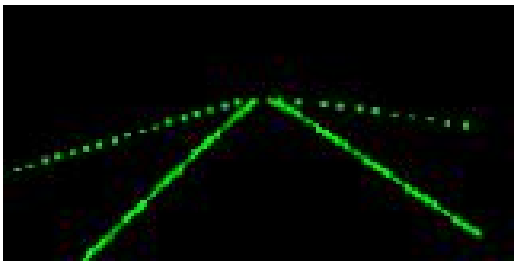
Figure 3.12: Frame and label 1 of first TuSimple dataset

(a) Frame 2 (b) Label 2

Figure 3.13: Frame and label 2 of first TuSimple dataset



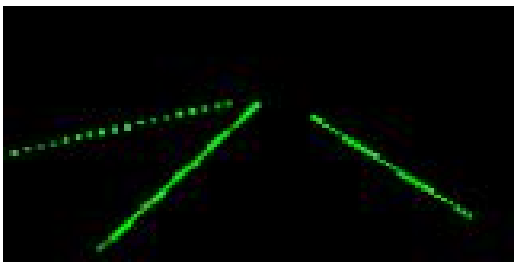
(a) Frame 3



(b) Label 3

Figure 3.14: Frame and label 3 of first TuSimple dataset

(a) Frame 4



(b) Label 4

Figure 3.15: Frame and label 4 of first TuSimple dataset

Second TuSimple Dataset

After creating the first TuSimple dataset and use it to train the firrstversion of the TuSimple model, due to the characteristics of the labels, the model performed poorly to detect the main road.

Since the main objective of the model is to detect the egolanes, and only this lanes, the next dataset was created only using the egolanes labels.

As mentioned before, the labels were retrieved from a json file in the form of the x-coordinates of each lane in a separate list in 'lines'. The first two items of the list 'lines are always the egolanes.

Selecting only the first to items of the list 'lines' at the moment of creating each label will come up with each label only containing the egolanes.

The same process is followed here to increment the number of pixels inside each label, and the final result can be seen in the 3.16.

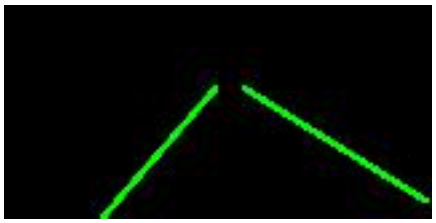


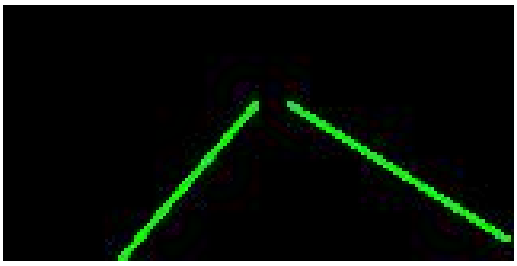
Figure 3.16: New label only showing egolanes

As seen in the figure, in this dataset there will be only two lines to be detected, the two egolanes.

To increase the size of the dataset, the same image transformation was applied to the frames and labels. The final dataset tha was created waas the TuSimple_egolanes_img_160x80.p and TuSimple_egolanes_lab_160x80.p, with a size of 406 Mb and 136 Mb respectively.

The following images represent a small preview of the second dataset created with the TuSimple data.

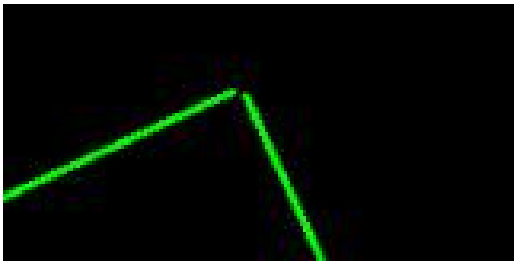
(a) Frame 1



(b) Label 1

Figure 3.17: Frame and label 1 of Egolane TuSimple dataset

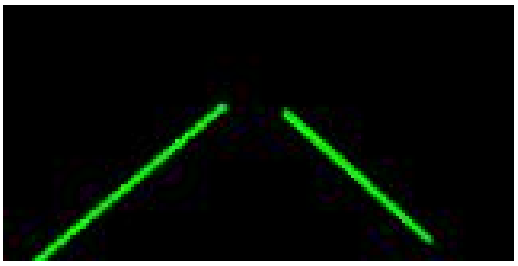
(a) Frame 2



(b) Label 2

Figure 3.18: Frame and label 2 of Egolane TuSimple dataset

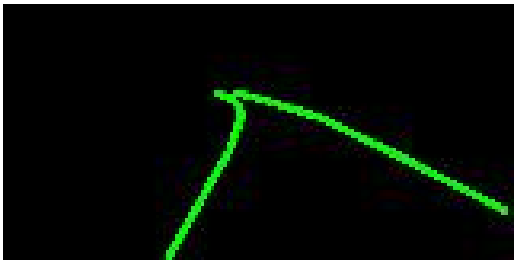
(a) Frame 3



(b) Label 3

Figure 3.19: Frame and label 3 of Egolane TuSimple dataset

(a) Frame 4



(b) Label 4

Figure 3.20: Frame and label 4 of Egolane TuSimple dataset

Another version of this dataset was created to train a new variation of the baseline model. The main difference of this new dataset is the size of each frame. Instead of a 160 by 80 pixel quality, the new dataset quality will be 320 by 180 pixels, increasing the image and label resolution.

3.3 Fuso Dataset

The third dataset used in the development of the models is a home made dataset using the MFTBC truck video already recorded in the Truck testing circuit facility. The main objective of creating this dataset is to retrain the model using the circuit images to increase the lane detection in the stone road, where previous models using the main two datasets were used, were not very effective to detect the road.

The source of this dataset, as previously mentioned, comes from real footage gather directly from the FUSO Testing Circuit. This videos were recorded with the camera attached to the truck during previous endurance tests. The videos were this dataset takes the images have been recorded during different hours, having footage with different lightning and even night footage.

The quality of the videos depends on when they were taken. The difference can be observed between the **Night Footage** and the **Day Footage**.

- **Day Footage:**: All videos recorded between 6am and 8 pm, corresponding to the Day period. The quality of these clips recorded during that period are 800x464 pixels.

- **Night Footage:** All videos recorded between 8 pm and 6 am, corresponding to the Night period. The quallity of these clips recorded during that period is 640x360 pixels.

The following Images are a representation of the **Day Footage** and the **Night Footage**.

Figure 3.21: Day Footage Example

Figure 3.22: Night Footage Example

This quality difference is not going to be determinant for the different models performance. Since all images that will be part of the datasets are going to be downsized to an image quality of 160x80 pixels to be fed into the model, having different quality image inputs will not alter the output.

To generate the dataset it is necessary to create the labels for each image. The dataset structure will be similar as the Baseline Dataset, consisting of a series of images having the whole road as a label, only having the Green channel as output.

Generating this new dataset from scratch will require to manually detect the road for each frame used in the dataset and save in in corresponding order. Since there is no real automatic option to generate the labels, the decision was to create a tool with which create all labels manually.

Labeling tool to create FUSO Dataset

To create the FUSO dataset and the Test dataset, a new labelling tool was created. This label tool is based on Python and it uses image transformation methods to create the dataset.

This labelling tool is run in a Jupyter Notebook using Python, and the main libraries used to create the labels are matplotlib and Open CV.

The process of the labelling tool to create all labels is the following:

The video where the images wants to be taken is opened, setting the exact frame to start taking the labels. Once the video reaches that exact frame, a new window appears, and manually by creating lines a polygon with the shape of the road is created. Once the polygon is created, the program saved both the frame and the label, which will consist of the polygon filled, only using the Green Channel. Then, the video goes to the next frame, which it can be also determined the distance between frames. The process is repeated until the video ends or by key interrupt. Both frame and label are saved in a reduced size, 160x80, having the same size as the initial Baseline dataset.

The following images represent the polygon creation and an example of the image created.

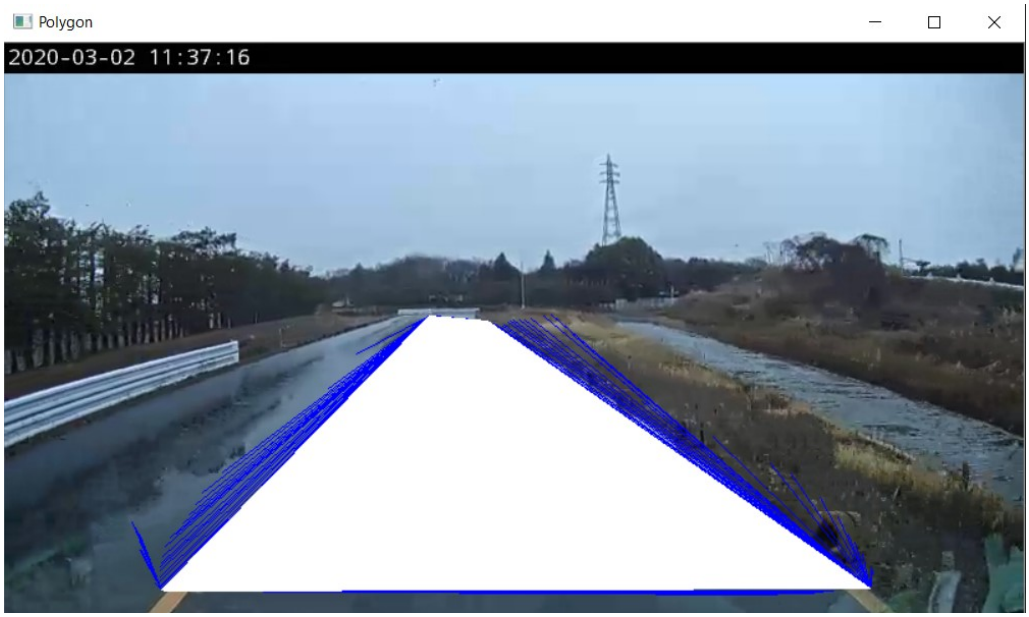


Figure 3.23: Labeling tool process: polygon creation

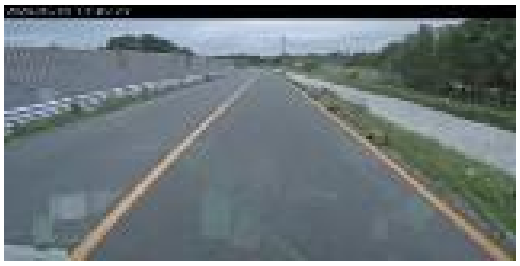
With this tool all labels used in the FUSO dataset were created.

FUSO dataset creation

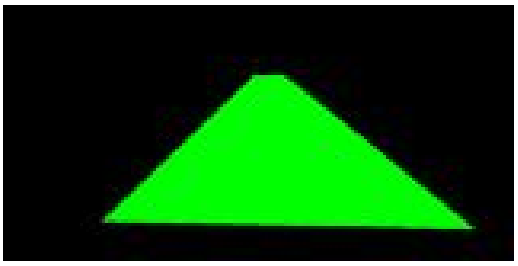
The total number of images taken were 168, taken from different videos, with different climate conditions and lightning. To increase the number of images in the dataset, image transformation was used. All images and labels were mirrored on the horizontal axis and tilted by 4^{o} degrees.

After this image transformation the final FUSO dataset consisted in 504 images and labels. This datset was used mainly to retrain the different models and to augment the already existing dataset to increase the model performance during the Rough Road prediction.

The following images are a representation of the FUSO dataset:



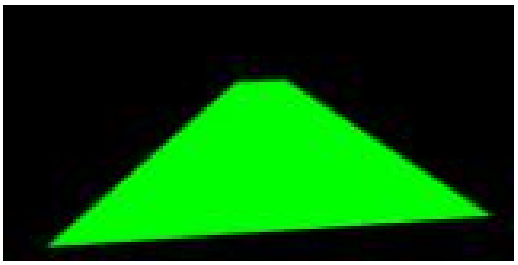
(a) Frame 1



(b) Label 1

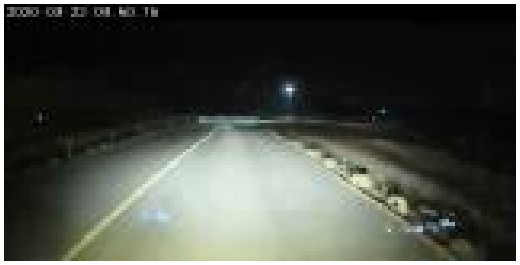
Figure 3.24: Frame and label 1 of FUSO dataset

(a) Frame 2

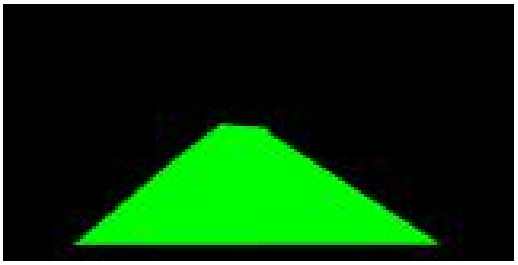


(b) Label 2

Figure 3.25: Frame and label 2 of FUSO dataset

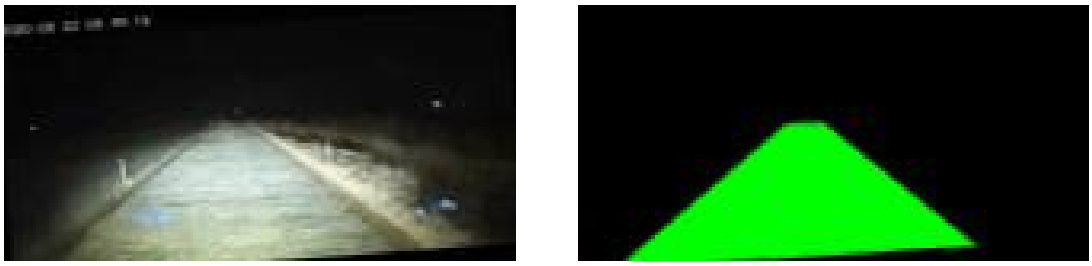


(a) Frame 3



(b) Label 3

Figure 3.26: Frame and label 3 of FUSO dataset



(a) Frame 4 (b) Label 4

Figure 3.27: Frame and label 4 of FUSO dataset

3.4 Test Dataset

The test dataset will be used to test all different models during the whole project develpment.

Since there are two types of models based on the two different types of dataset, there should be two different test dataset, one for each type of dataset.

For the TuSimple dataset, which models will only detect the two main sidelines of the road, the test dataset will consist of a series images from the MFTBC video images of the Truck Test Circuit. All labels were generated with the labeling tool already explained in the FUSO dataset section. The labels of the test images will consist of a pixel map of the egolines, similar to the labels from the TuSimple dataset, but instead of a blurry pixel cloud, which consisted of a pixel map of two polygons.

For the Baseline Dataset and the Fuso Dataset, the models based on these dataset have as output the whole road, instead of only the sidelines of the road. Therefore the test dataset will consist of a series images from the MFTBC video images of the Truck Test Circuit, being the labels same as the training datasets. The labels of the dataset will consist of one poligon pixelmap with the shape of the road.

The images used to create this dataset were selected from different videos with different lightning and weather conditions. The summary of the images can be shown here:

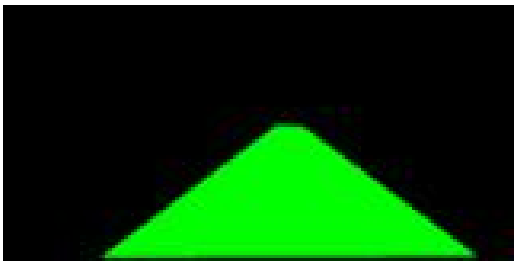
- 25 images taken from a Clear Day video.
- 10 Images taken from a Clear Night video.
- 10 images taken from a Rain Day video.
- 10 images taken from a Fog Day video.
- 15 images taken from a Sunrise video.
- 10 images taken from a Rain Night video.

All images will be tilted and mirrored horizontally to increase the size of the dataset and have a more distributed result of the test. The final Test Dataset consisted on 240 images and 240 labels, saved in different Pickle files to be used to test each model.From all 240 images, 80 were original images, 80 were the result of tilting each original image and its label by 4 degrees and 80 were result of mirroring the original images and their labels.

This Test Dataset will be used to measure the Loss and the Accuracy of each model that uses the Virgo Dataset or the Fuso dataset.

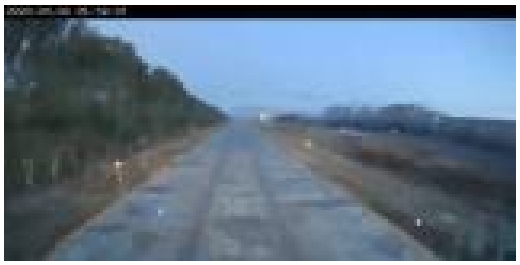
The following images show four examples of the Test Dataset

(a) Frame 1

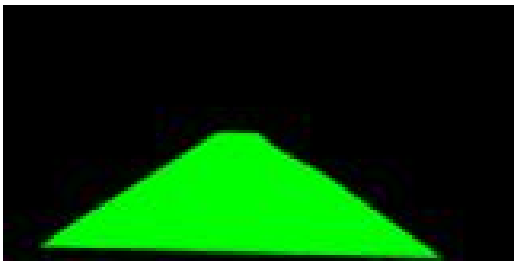


(b) Label 1

Figure 3.28: Frame and label 1 of Test dataset



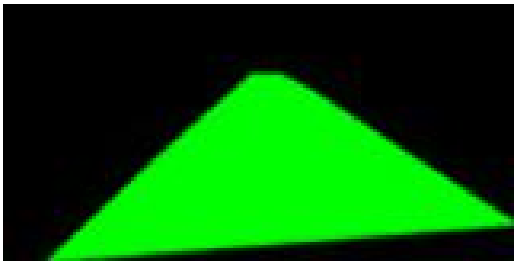
(a) Frame 2



(b) Label 2

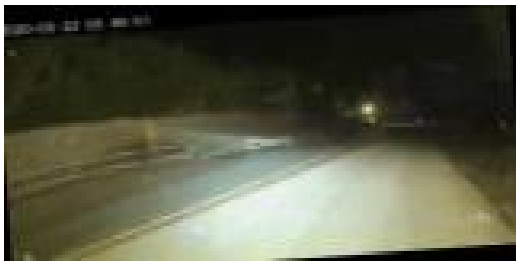
Figure 3.29: Frame and label 2 of Test dataset

(a) Frame 3

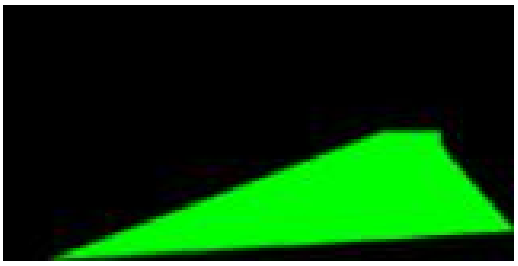


(b) Label 3

Figure 3.30: Frame and label 3 of Test dataset



(a) Frame 4

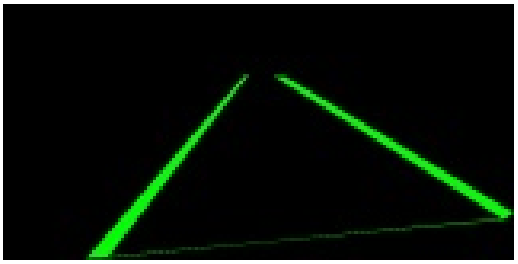


(b) Label 4

Figure 3.31: Frame and label 1 of Test dataset

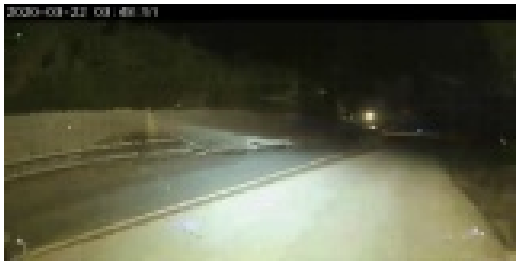
Here are four examples of the TuSImple Test Dataset, which will be used to test all TuSimple based models.

(a) Frame 1

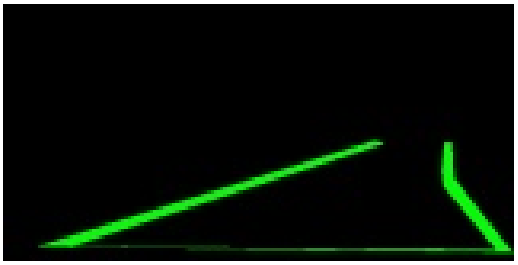


(b) Label 1

Figure 3.32: Frame and label 1 of TuSImple Test dataset

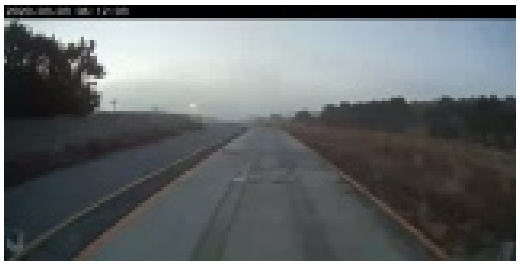


(a) Frame 2

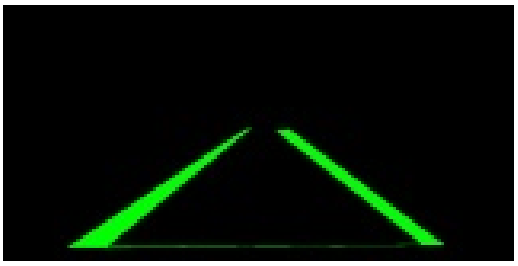


(b) Label 2

Figure 3.33: Frame and label 2 of TuSImple Test dataset

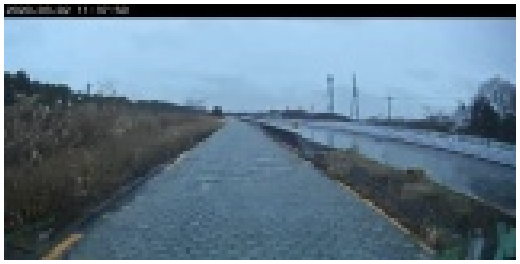


(a) Frame 3

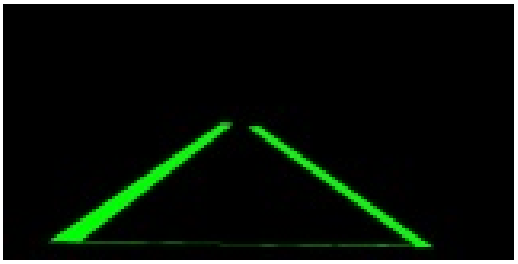


(b) Label 3

Figure 3.34: Frame and label 3 of TuSImple Test dataset



(a) Frame 4



(b) Label 4

Figure 3.35: Frame and label 4 of TuSImple Test dataset

Chapter 4

Models Implementation

This chapter will include all different models tried and a description of them. For each model mentioned there is a summary of the model architecture, with a table representing the neural network distribution and all different layers and the number of parameters.

All models were created in Python, using the Keras library to develop the different Neural Networks. Keras is an open source library used to build models which are based in neural networks. The Sequential method was used to create all the diferent layers and

Besides the structure of the Neural network, each section will include a brief description of the results taken with each model. Each model will be tested using different methods. First, each model will show the output result on different scenarios of the circuit, showing the output of the model during Day, Night, rain and between the Smooth road and the Rough road.

4.1 SegNet Baseline Model

The Baseline model is based on the SegNet architecture [6]. SegNet is a deep encoder-decoder architecture for multi-class pixelwise segmentation researched and developed by members of the Computer Vision and Robotics Group at the University of Cambridge, UK.

The model uses the encoder-decoder architecture to create a binary segmentation of the road. Each pixel will be determined if they are part of the lanes or not. The structure of the neural network consist exclusively on convolution layers and pooling layers. The first layer of the model is a Batch Normalization layer to speed up the model training and running. After the Batch Normalization layer, the model structure follows a mirrored sequence of Convolution-deconvolution.

The model consist in a sequence of convolution layers with RELU activation, followed by a Pooling layer to reduce the size of the amount of parameters needed. It uses Max Pooling to select the maximum values of the previous layer in each kernel. This process is then copied again, using convolutional layers and polling layers.

After this Convolution + Pooling process is finished, a mirrored version of Upsampling and Deconvolution is created to recreate the output pixel map segmentation to the same size as the input image. To increase the robustness of the whole model, inside the inner Convolution layers Dropout is introduced.

Dropout consist in ignore a percentage of units in the layers for each iteration of the model. This will help to create a more robust model and to prevent over-fitting the neural network.

The final model structure is represented in the following table:

Layer (type)	Output Shape	Param
batch normalization 1 (Batch)	(None, 80, 160, 3)	12
Conv1 (Conv2D)	(None, 78, 158, 8)	224
Conv2 (Conv2D)	(None, 76, 156, 16)	1168
max pooling2d 1 (MaxPooling2	(None, 38, 78, 16)	0
Conv3 (Conv2D with dropout)	(None, 36, 76, 16)	2320
Conv4 (Conv2D with dropout)	(None, 34, 74, 32)	4640
Conv5 (Conv2D with dropout)	(None, 32, 72, 32)	9248
max pooling2d 2 (MaxPooling2	(None, 16, 36, 32)	0
Conv6 (Conv2D with dropout)	(None, 14, 34, 64)	18496
Conv7 (Conv2D with dropout)	(None, 12, 32, 64)	36928
max pooling2d 3 (MaxPooling2)	(None, 6, 16, 64)	0
up sampling2d 1 (UpSampling2)	(None, 12, 32, 64)	0
Deconv1 (Conv2DTranspose with dropout)	(None, 14, 34, 64)	36928
Deconv2 (Conv2DTranspose with dropout)	(None, 16, 36, 64)	36928
up sampling2d 2 (UpSampling2)	(None, 32, 72, 64)	0
Deconv3 (Conv2DTranspose with dropout)	(None, 34, 74, 32)	18464
Deconv4 (Conv2DTranspose with dropout)	(None, 36, 76, 32)	9248
Deconv5 (Conv2DTranspose with dropout)	(None, 38, 78, 16)	4624
up sampling2d 3 (UpSampling2)	(None, 76, 156, 16)	0
Deconv6 (Conv2DTranspose)	(None, 78, 158, 16)	2320
Final (Conv2DTranspose)	(None, 80, 160, 1)	145

Table 4.1: Full structure of the model

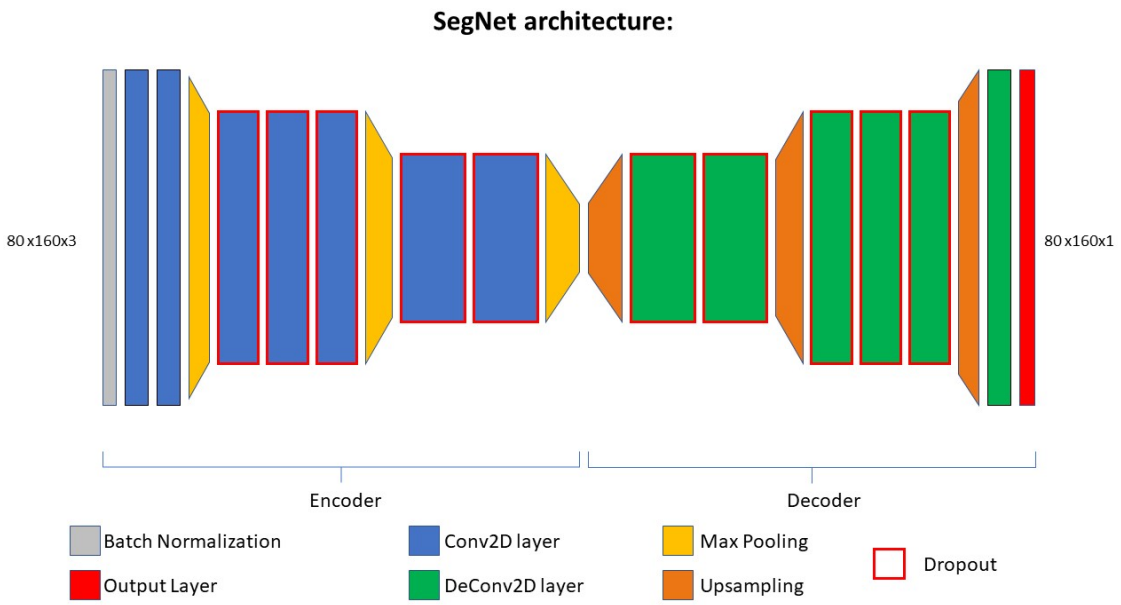


Figure 4.1: Baseline model architecture

A visual representation of the structure of the model can be seen in the following image:

The training data used to fit the model was the Baseline Dataset. Tgis dataset is explained in Chaptere 3, and consist of 1276 images and its labels of the whole road.

The output from this model is a one-channel image of the pixel segmentationmap of the prediction of the road, with same size as the Training dataset used. In this case the Input image is resized to 160 x 80 pixels before being fed into the model. After the output is transformed to a RGB Image, with only the Green channel filled, the output image is resized to the original dimensions.

Then, the output image and the input image are merged to show a real representation of the pixel segmentation map compared to the road.

This section will be dedicated to explain how the base model was tested and how good the prediction is, and how accurate the output pixel map identifies the road.

To see how the model output behaves, the first try was to feed the model with single images, getting just another image as output.

The following images were taken from videos already recorded from the camera installed in the truck, previous to the model implementation. The images were taken form different videos and at different hours to determine how good the model is capable to detect the road with different lightning. Also, the images are taken in different stages of the circuit to see how the model behaves in the two different road types, the Smooth Road

and the Rough Road.

(a) Input Image (b) Merged Output

Figure 4.2: Predicted image in Smooth Road at Clear Day with Marked Lines

(a) Input Image (b) Merged Output

Figure 4.3: Predicted output in Smooth Road with no Marked Lines

(a) Input Image (b) Merged Output

Figure 4.4: Smooth Road curve prediction

(a) Input Image (b) Merged Output

Figure 4.5: Rough Road with no marking lines prediction

This images were taken from a video taken in a Day Clear. In each of the four figures the input and the merged output images are shown. The input image on the left and the output on the right. Each input image has a image resolution of 800 x 464 pixels. The output was re-scaled to this resolution after it was processed by the model.

In the images shown it is possible to see that the model, if the road has a good asphalt and delimited lines with good visibility, can obtain good results detecting the main road. It is not completely precise and some parts of the road are not detected, but the model identifies both EgoLanes and is capable of represent the road (Figure 4.2 and Figure 4.3)

However, when the model tries to detect the road with stone pavement, the model is not capable of create a good pixel segmentation map (Figure 4.4 and Figure 4.5). The output pixel segmentation map is non existing in this type of road and the model would not be able to detect any possible deviations in this part of the road This issue can be caused due to a lack of similar road type in the training dataset.

After testing the model with daylight images, the next images tested were in different lightning conditions. The images selected were taken from the same camera truck at different hours. The following images represent the night footage chosen to test the model at night and sunrise. These night images were also taken to represent different stages of the circuit and with different road types.

The next images were taken from the camera located in the truck. As mentioned before, they represent different locations on the circuit. The road is illuminated artificially using the truck's headlights. These light make more difficult to perceive the lines at the edge of the road from the main road

(a) Input Image (b) Merged Output

Figure 4.6: Night footage of the Smooth road

(a) Input Image (b) Merged Output

Figure 4.7: Night footage of the Rough road

The results are very similar to the ones obtained with regular daylight. The model is capable to detect properly the main road if it is asphalt. Same as the previous footage from daylight, the model performs poorly trying to detect the road when the pavement is made by stones.

The last to images were taken at dawn, with low light but without using the truck headlights.

(a) Input Image (b) Merged Output

Figure 4.8: Sunrise footage of the Smooth road

(a) Input Image (b) Merged Output

Figure 4.9: Sunrise footage of the Rough road

The footage from sunrise lightning has the best output from all three different light settings. The model in this scenario is capable to the detect the road in both asphalt and stone roads.

Besides these images video recordings have been tested and the results were very similar to the ones obtained with only images. The model has issues to detect the stone road in all different light settings and it is not completely accurate during the whole circuit.

4.2 TuSimple Baseline Model

Another proposed way to approach the Line Detection problem was, instead of detecting the whole road, the model will only detect the EgoLanes of the road, and therefore, the limits of the road. This approach was based on the TuSimple dataset which used as labeled ground truth only the lines of the road, and not the whole road. This TuSimple Baseline model is the first iteration of this second approach to the Lane Detectin problem, and will serve as baseline for all Egolane Detection models.

The TuSimple Baseline model shares the same Neural Network Structure as the Basline Model. It is based on SegNet architecture and is a Encoder-Decoder architecture where the Decoder is a mirrored version of the Encoder.

The full structure of the TuSimple Baseline Model is shown in the table 5.6.

This model used the TuSimple_egolane datset as training dataset. This dataset, already described in Chapter 3 consist in a total of 14504 images. This images are labeled only with the EgoLanes, and the output will be a pixel segmentation map but only of the EgoLanes, not the whole road as the Baseline model. The accuracy and the loss validation of this model will not be comparable to the loss validation and accuracy if the baseline model and this model will serve as baseline for all TuSimple dataset based models.

Same as the Baseline model, the input image was resized to the same size as the Imga size of the TuSimple Dataset. The input image size fed into the model was 160 x 80 pixels, and the output has the same size. After the model gives the output, it is transformed into an RGB image and merged with the input image and resized to their real size.

4.2.1 Model Testing and Output Behaviour

This section will be dedicated to explain how the model behaves in different road configurations and to test how accurate the model detect the lines.

The first way to test the models output behaviour was to feed single images and see how the pixel segmentation map which was the output was able to detect the EgoLanes.

The following images show the output from the TuSimple Baseline model already merged beside the input image in different road and lightning scenarios.

Clear Day:

(a) Input Image (b) Merged Output

Figure 4.10: Predicted image in Smooth Road at Clear Day with Marked Lines

(a) Input Image (b) Merged Output

Figure 4.11: Predicted output in Smooth Road with no Marked Lines

(a) Input Image (b) Merged Output

Figure 4.12: Predicted image in curve situation

(a) Input Image (b) Merged Output

Figure 4.13: Rough road with no marking lanes

All four input images were taken from a clear day video footage. It can be appreciated that the model output performance in optimal circumstances (Smooth Road and Marked Lines) 4.10 is very good. It detects the left line perfectly and it has some issues detecting the right egolane. This can be due to the difference between the pavement and the grass.

In the other three figures the output performance is not as good as in the Optimal configuration. In 4.11 the output performance is similar to the optimal configuration, being able to detect the left egolane but has some trouble detecting the right egolane. In 4.12 the model has problems to detect the curve line properly and is not able to detect the right lane at all. In 4.13 it is possible to see that the output is almost in-existent. It is possible to see false positives inside the road that are not part of the two main EgoLanes. This can be caused due to the difference between the type of the road from the dataset and the type of the road from the input image.

Night Clear:

(a) Input Image (b) Merged Output

Figure 4.14: Smooth Road at Night prediction

(a) Input Image (b) Merged Output

Figure 4.15: Rough Road at Night prediction

For Night footage, the output performance of the model is not good. Similar to the baseline model, it has problems detecting the Rough Road, in a Smooth Road it can detect the left line almost perfect, but as seen in the Clear Day footage, it has trouble detecting the right lane. As for the Rough road input, with the artificial light from the truck lights, the pavement stone is not as interfering as with daylight. The model still detects some false positives in the middle of the road, and is not able to detect the right egolane, but it detects the left egolane.

Sunrise:

(a) Input Image (b) Merged Output

Figure 4.16: Smooth Road at Sunrise prediction

(a) Input Image (b) Merged Output

Figure 4.17: Rough Road at Sunrise prediction

The Sunrise footage output performance seems to have the best results. Both EgoLanes are detected properly in both Smooth and Rough road scenarios.

Testing TuSimple Model with TuSimple Test Dataset

To determine the performance of the Egolane Detection models not only using the Output images, these models will be tested using the same technique as the Whole Road Detection models.

The dataset used to test these models is the TuSimple Test dataset, which is explained in Chapter 3. This dataset consist of 240 images and the labels of the EgoLanes of the circuit road in different environments.

This method will use the model already loaded using the model.load method and evaluate it measuring the Accuracy and the Loss. This baseline model accuracy and Loss will be used as the starting measurements and it is expected for future models to have better results in this evaluation test.

The results obtained from evaluating the TuSimple Baseline model are the following:

4.3 Retrained models

After the first results obtained from the Baseline model, the main insights extracted from them is that the model perform really poorly in Rough Road situation, in all different lightning scenarios. This poor performance is mainly caused by the difference between the training data road pavement and the MFBTC FUSO Truck Testing facility circuit road type.

To try get rid of the Rough Road problem the next step chosen was to retrain the Baseeline Model with some data extracted from the MFBTC FUSO Truck Testing facility circuit video footage already existing. This data was used to create the FUSO dataset, which consist of 504 images and labels.

The first attempt of retraining the existing model was to create a new final layer and train only the last layer with the FUSO data. The results obtained from that model were really not good. The output representation of the Retrained_model_1 was worse than the Baseline model, and the decission was to not continue adding and training more layers in the model.

The next approach to retrain the model was to use the FUSO dataset and Tune Fit the Baseline model, decreasing the learning rate of the model to a very low value, to not overfit the model.

The process of Tune Fitting the Baseline model was done using Keras [13]. The Baseline model was loaded and compiled again using the method model.compile, using Adam as optimazer and Mean Square Error as loss measurement.

Several models were Retrained using this method, specifying the learning ratio to different values and changing the train dataset size, the epochs and the batch size durint the Tune Fitting.

The following table show the different models retrained and their specifications:

Retrain model specification

Model	Adam's learning rate	Datset Size	Epochs
Retrained_2	e^{-6}	200	20
Retrained_3	e^{-5}	200	20
Retrained_4	e^{-6}	300	25
Retrained_5	e^{-5}	504	20
Retrained_6	e^{-6}	504	20

Table 4.2: Baseline model test metrics

For all different models Retrained, they were tested using the same video and footage and the best performing model from them all was the Retrained_5. This model was chosen

to be representative for the Retrained models, since it was the model with the best output performance overall.

The following images show the output representation of the Retrained_5 model:

(a) Input Image (b) Merged Output

Figure 4.18: Predicted image in Smooth Road at Clear Day with Marked Lines

(a) Input Image (b) Merged Output

Figure 4.19: Predicted output in Smooth Road with no Marked Lines

(a) Input Image (b) Merged Output

Figure 4.20: Predicted image in curve situation

(a) Input Image (b) Merged Output

Figure 4.21: Rough road with no marking lanes

The four images above show the input and merged output of a Clear Day by the Retrained_5 model. It can be noticed that the output in all input images has a good performance and it detects the road properly in all circumstances. Even in the figure 4.21, where the Rough road is acting, the model is able to predict the road with good results.

This output representation is not reliable at 100%. During the video output testing, the Retrained_5 model performed better than the Baseline model, but still,it had moments, always in the change between Smooth road and Rough road, the model lost track of the output and was not able to detect the Rough road properly between 100 and 120 frames after the change.

Sunrise:

(a) Baseline output (b) Retrained Output

Figure 4.22: Smooth Road at Sunrise prediction

(a) Baseline output (b) Retrained Output

Figure 4.23: Rough Road at Sunrise prediction

The Sunrise output performance in both Smooth and Rough road situations is good. The predicted pixel-map covers almost the whole road.

One problem that has been observed during the video testing of the Retrained model was the big amount of false positives that the retrain model outputted. In many frames, the model detected as road parts of the field, and therefore, not being able to differentiate the margins of the road.

4.4 Models Behavior with Image Transformation

After testing the models, it is clear that all models have issues trying to detect the road in Rough Road scenarios. As mentioned before, this is due to the difference between the road type from the Train datasets, where all images roads are made from asphalt and the pavement is smooth, without substantial changes in the road besides the lanes and some defects that are not very frequently. On the other hand, the Rough Road from the MFTBC Truck Testing Facility Rough road is made of paving stone. This paving stone road image has different color gamma than the asphalt road, and the models doesnt detect this type of road well.

To mitigate the Rough Road effect on the detection process, it was decided to try some image transformation methods to the input before trying new Neural Networks architectures.

The images transformation methods used were image blurring and image smoothing. These methods took the input image and blur all parts of the image to create a more smooth view, eliminating the paving stone tessellation effect and making the Rough Road to be more similar to the Smooth Road.

To implement this image transformation methods, the library used was OpenCV for Python, and this transformation was implemented just before the image is fed to the model. The methods used to smooth the images are the following:

- **Averaging**: "Averaging simply takes the average of all the pixels under kernel area and replaces the central element with this average. This is done by the function cv2.blur()" [14]. This function requires as input the image which is going to be transformed and the size of the kernel. Blurring the image is the basic way to smooth the road surface and make it more homogeneous. The following images show a Rough road Input Image before and after the blurring:

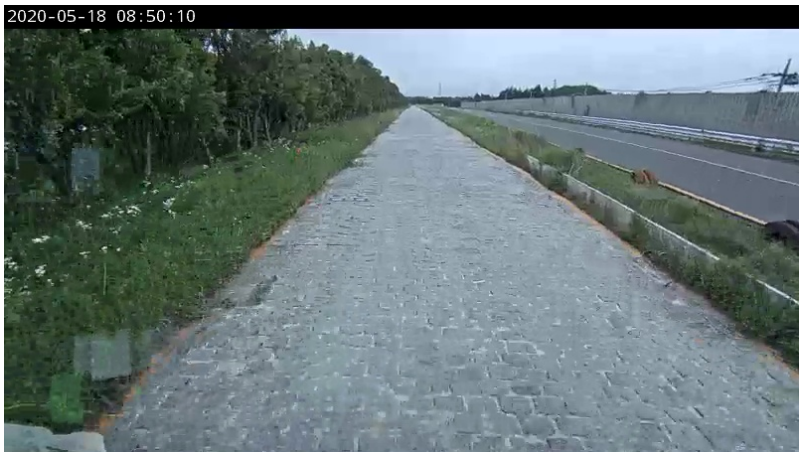


Figure 4.24: Input image before blurring

Figure 4.25: Input image after blurring

- **Median Filtering**: Median filtering works similar to Averaging but instead of using the average of the pixels, it computes the median of the pixels inside the kernel and replaces the center with it. The following images are a representation of the Median Filtering:

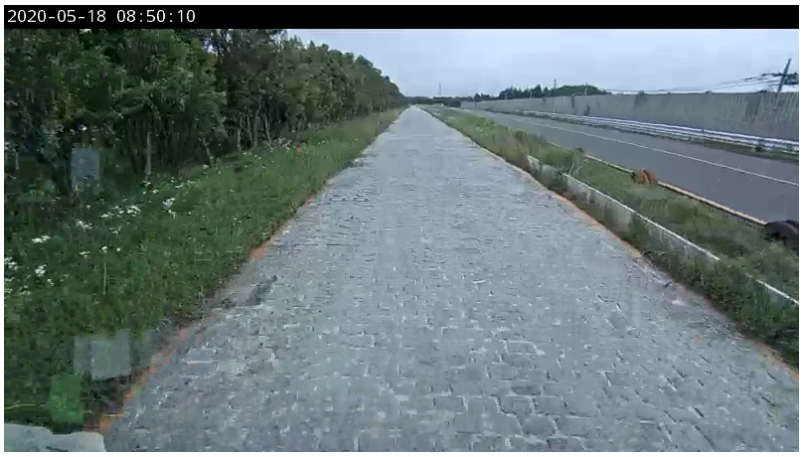


Figure 4.26: Input image before Median filtering

Figure 4.27: Input image after Median Filtering

- **Bilateral Filtering**: Blurring method that uses same method as Averaging, but Bilateral Filtering does not blur the edges of the image, keeping intact the edges of the road and only smooth the inside of the road. "The bilateral filter also uses a Gaussian filter in the space domain, but it also uses one more (multiplicative) Gaussian filter component which is a function of pixel intensity differences" [14]. The function is cv2.bilateralFilter(), and it has as input the image wanted to be filtered, the size of the kernel and the size of the Edge detection window.The following images are a representation of the Bilateral Filtering:

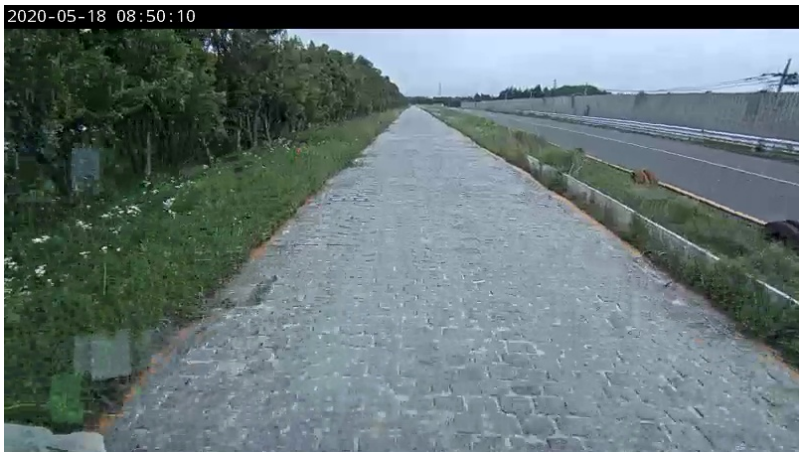


Figure 4.28: Input image before blurring

Figure 4.29: Input image after blurring

The first attempts using Image Smoothing were using only Averaging in the images, but the predicted output did not experience a noticeable improvement compared to the original models outputs. Different levels of blurring were tested, but the problem encountered was that if the blurry level was low (kernel size 5), the smoothing of the Rough road was not enough smoothed for the model to perceive any difference. If the blurring level was too high (kernel size 15 above) the image was blurred in a way the edge of the road mixed with the surroundings, and the model was not able to differentiate between the road and the field, resulting on a bad detection.

The following image shows the output of a Clear Night output prediction from the Baseline model where the input image was blurred wit a kernel size of 25.

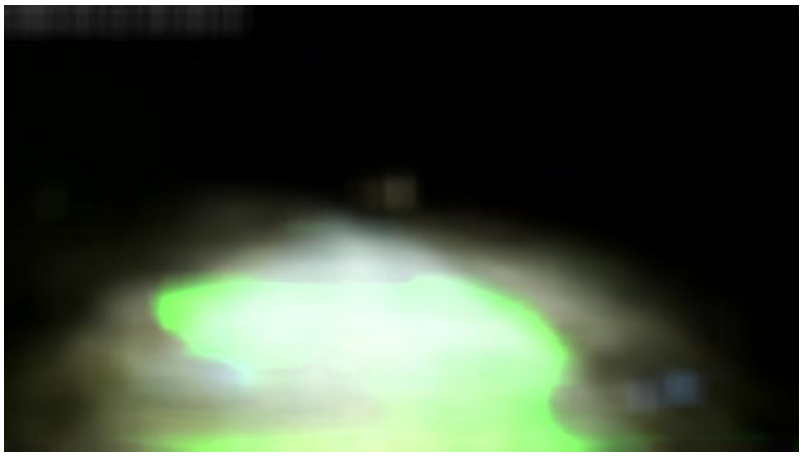


Figure 4.30: Output from overblurred image

It is possible to notice that the margins of the road are not noticeable and therefore the model is not able to detect the road.

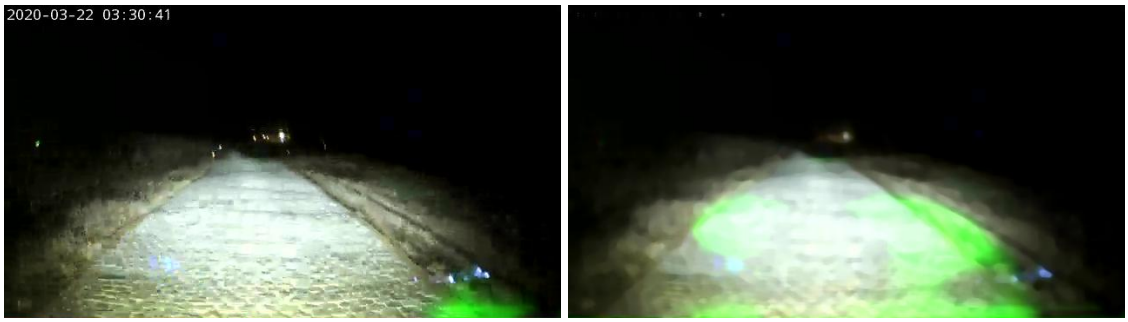
However, using the Bilateral Filtering will achieve to blur the road and preserve the edges of it.

The best approach using image smoothing after several experiments was combining the Bilateral Filtering with the Median Average. First a kernel size 9 Bilateral Filter is used to smooth the road and keep the edges, and then a kernel size 9 Median average increases the road smoothness.

The results compared to the non smoothed input image are the following:

(a) Baseline without smoothing (b) Baseline with smoothing

Figure 4.31: Baseline output comparison with smoothing

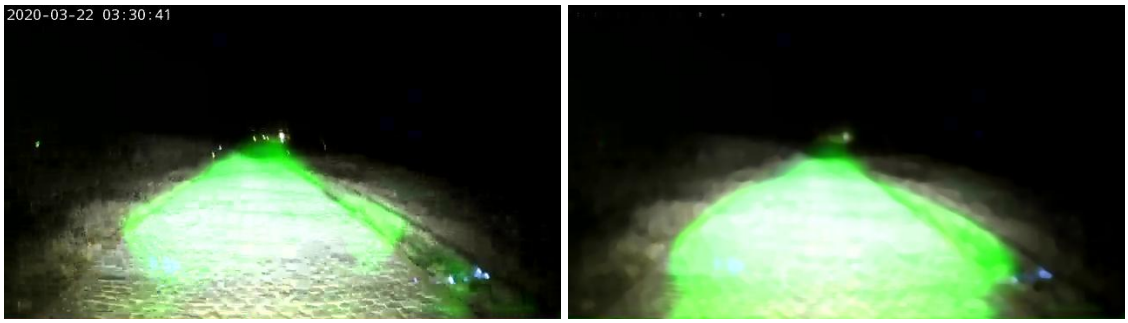


(a) Baseline without smoothing (b) Baseline with smoothing

Figure 4.32: Baseline night output comparison with smoothing

(a) Retrained_5 without smoothing (b) Retrained_5 with smoothing

Figure 4.33: Retrained_5 output comparison with smoothing



(a) Baseline without smoothing (b) Baseline without smoothing

Figure 4.34: Retrained_5 night output comparison with smoothing

The results obtained during the image transformation suggest that smoothing the road has an observable impact in the Baseline model performance in day, Figure 4.31, but in

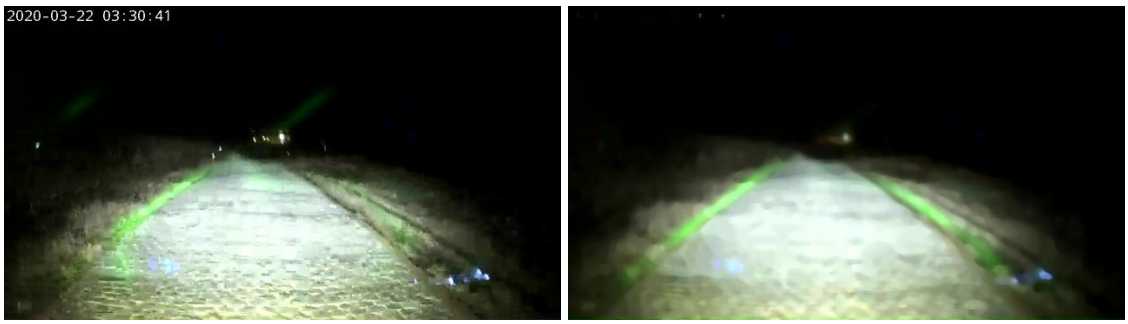
night scenarios the output still performs very poorly, as shown in figure4.32. For the Retrained_5 model, since it was retrained with the circuit data, the output in Clear Day was already good, and the difference between smooth and not smooth image is not appreciable, as seen in Figure 4.33. However, in night scenario, the output of the smoothed image is slightly better, with the output pixel segmentation map covering a bigger area of the road as seen in figure 4.34

Although the Image Smoothing has showed a slight improvement in both models performance, being the Baseline model the most affected of them all, the Bilateral filtering requires computational time that makes not viable to apply these Image smoothing in a real scenario which is the final objective of the models.

For the TuSimple Baseline model the image smoothing did not have a noticeable change in the output performance. The smoothed image has less false positives in the center of the road, but the quantity of pixels detected in the pixel map is still very reduced compared to the real road. The following images shows a comparison of the TuSimple Baseline model with and without Image Smoothing:

(a) TuSimple Baseline without smoothing (b) TuSimple Baseline with smoothing

Figure 4.35: Retrained_5 night output comparison with smoothing



(a) TuSimple night without smoothing (b) TuSimple night with smoothing

Figure 4.36: Retrained_5 night output comparison with smoothing

4.5 ERFNet based models

After trying image smoothing as a way to increase the performance of the SegNet based models, the results obtained showed that the output performance did not change in a noticeable way. The next step was to try new methods to detect the lane.

After the SegNet approach, a new Neural Network architecture was tried. The architecture chosen for the next developed models was the ERFNet architecture. Taking the ERFNet architecture as a reference, the proposed new model architecture has a similar structure, but all layers of the model consist of 2D Convolutional layers instead of 1D Convolutional layers.

This model uses the same layers layout as the ERFNet architecture.The proposed new model architecture has a similar structure, but all Convolutional layers of the model are 2D Convolutional layers instead of 1D Convolutional layers, as the original ERFNet has. This change in the structure of the layer itself will result in a big increase of the computational power needed by the model , and therefore a big reduction the FPS that the model will be able to process.

4.5.1 ERFNet Full architecture

The first model having the ERFNet as a reference has the following structure:

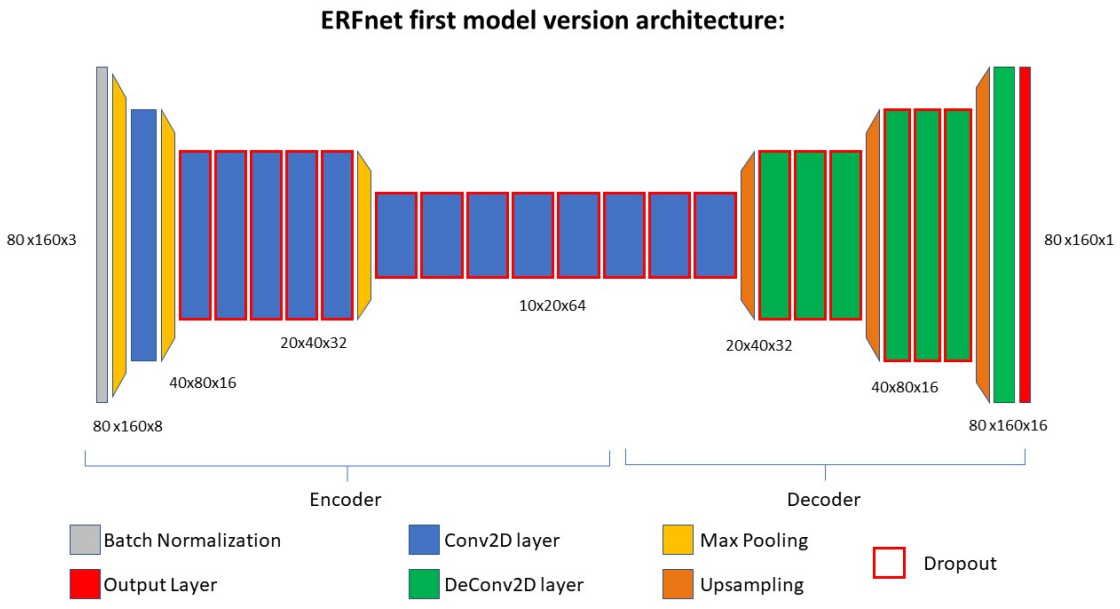


Figure 4.37: First ERFNet based model

This model consist of a combination of 2DConvolutional layers and max pooling layers which form the Encoder structure, and a series of 2D Deconvolutional layers and upsampling layers forming the Decoder structure. All Convolution and Deconvolution process have the ReLU function as activation function, and all layers use padding to maintain the same input size along the whole Neural Network.

The first layer of the Neural network, same as used in the SegNet architecture, is a batch normalization layer. Right after the normalization a max pooling layer downsizes the input image and passes it to the firs Convolutional layer of input size (40, 80, 3). After this first convolution another max pooling is done, followed by five 2D Convolutional layers with 32 kernels each, and an input size of (20, 40) and all of the layers after the second max pooling will have dropout to avoid overfitting of the model during the training stage of it. The last part of the decoder, the inner layers, consist of eight 2D Convolutional layers with 64 kernels each. The input size of these inner layers is (10,20) and their focus is to get all the deep insights from the image. All these inner layers will have dropout of 0.2 to avoid overfitting the model.

The decoder structure consists of differentiate three parts divided by the upsampling layers. Right after the inner layers part of the decoder a upsampling layer with a pool size of (2,2) leads to three Deconvolutional layers, another upsampling followed by another three deconvolutional layers. These Deconvolutional layers are have the exact input size as the Convolution layers. The last two layers of the Decoder are a Deconvolutional layer with 16 kernels right before the Final layer, which has the output size of (80, 160,1).

A more profound explanation of all the layers that forms the ERFNet based model is showed in the following table:

Only one type of model was created using this architecture. The model created using this architecture was trained using a new dataset, formed by the Baseline dataset plus the FUSO dataset, containing a total of 13268 images. The training parameters used to train the model are the following: batch_size per training iteration = 128, epochs = 25, pool_size = (2, 2).

The results obtained with the first ERFNet based model, the Full_ERFNet were very good, being able to detect the road in a accurate way in almost any scenario. But in some cases, the prediction of the model was not very precise, having in several cases high values of false positive pixels in the pixel segmentation map output.The model detected as road elements out of the limits of it, such as the field and even trees.

The following images shows an example of a False predicted output from the ERFNet based model.

Layer (type)	Output Shape	Parameters
batch_normalization_1 (Batch)	(None, 80, 160, 3)	12
max_pooling2d_1 (MaxPooling2)	(None, 40, 80, 3)	0
Conv2 (Conv2D)	(None, 40, 80, 16)	448
max_pooling2d_2 (MaxPooling2)	(None, 20, 40, 16)	0
Conv3 (Conv2D with Dropout)	(None, 20, 40, 32)	4640
Conv4 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
Conv5 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
Conv6 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
Conv7 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
max_pooling2d_3 (MaxPooling2)	(None, 10, 20, 32)	0
Conv8 (Conv2D with Dropout)	(None, 10, 20, 64)	18496
Conv9 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv10 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv11 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv12 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv13 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv14 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv15 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv16 (Conv2D with Dropout)	(None, 10, 20, 64)	3692
up_sampling2d_1 (UpSampling2)	(None, 20, 40, 64)	0
Deconv1 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	18464
Deconv2 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	9248
Deconv3 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	9248
up_sampling2d_2 (UpSampling2	(None, 40, 80, 32)	0
Deconv4 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	4624
Deconv5 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	2320
Deconv6 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	2320
up_sampling2d_3 (UpSampling2)	(None, 80, 160, 16)	0
Deconv7 (Conv2DTranspose)	(None, 80, 160, 16)	2320
Final (Conv2DTranspose)	(None, 80, 160, 1)	145
Total Number of parameters:		404,701

Table 4.3: Full structure of the model

Figure 4.38: ERFNet output prediction in normal conditions

Figure 4.39: ERFNet Fasle positive output prediction 2. It is possible to appreciate the model detects the road fence as part of the road.

In these two representation, Figure 4.38 and Figure 4.39, it is possible to appreciate how the model, even though it detects the lane very accurately in a Smooth straight case scenario, during the Sharp turns, there is a wide area of the pixel segmentation map which is categorized as a false negative. These low performance of the model appears always

in the same areas of the circuit, where the lines of the road are not well defined near the sharp turns.

The overall performance of the model is good and sufficient to detect the road in almost any scenario. Here is the results after testing quantitative the ERFNet Full model:

ERFNet based model Quantitative Test

Model	Precision	Recall	IoU	F1
ERFNet_full Model	0.8936	0.9564	0.8593	0.9217

Table 4.4: ERFNet model test metrics

4.5.2 ERFNet Reduced architecture

However, after some experiments changing the architecture of the model, a new model, with the same structure and the same max pooling layers, but reduced number of hidden layers in the inner levels of the Neural network. The new reduced ERFNet based model structure includes a new Convolutional layer just before the first max pooling, at the beginning of the Neural Network, to capture the first insights of the image at max resolution., and reduces the amount of inner layers after the second and third max pooling (from 5 2DConv to 4 2D Convolutional layers in the second max pooling and from 8 to 5 layers after the third max pooling).

The structure of the Decoder has remained the same as the ERFNet full model.

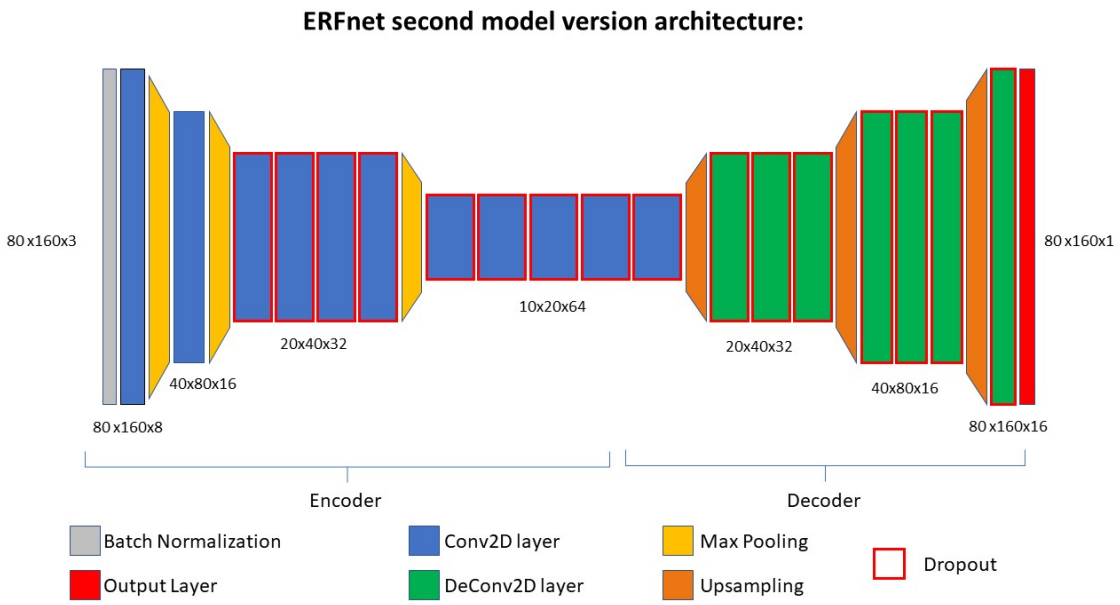


Figure 4.40: First ERFNet based model

A more profound explanation of all the layers that forms the Reduced ERFNet based model is showed in the following table:

Layer (type)	Output Shape	Parameters
batch_normalization_1 (Batch)	(None, 80, 160, 3)	12
Conv2 (Conv2D)	(None, 80, 160, 16)	224
max_pooling2d_1 (MaxPooling2)	(None, 40, 80, 3)	0
Conv2 (Conv2D)	(None, 40, 80, 16)	1168
max_pooling2d_2 (MaxPooling2)	(None, 20, 40, 16)	0
Conv3 (Conv2D with Dropout)	(None, 20, 40, 32)	4640
Conv4 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
Conv5 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
Conv6 (Conv2D with Dropout)	(None, 20, 40, 32)	9248
max_pooling2d_3 (MaxPooling2)	(None, 10, 20, 32)	0
Conv7 (Conv2D with Dropout)	(None, 10, 20, 64)	18496
Conv8 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv9 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv10 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
Conv11 (Conv2D with Dropout)	(None, 10, 20, 64)	36928
up_sampling2d_1 (UpSampling2)	(None, 20, 40, 64)	0
Deconv1 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	18464
Deconv2 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	9248
Deconv3 (Conv2DTranspose with Dropout)	(None, 20, 40, 32)	9248
up_sampling2d_2 (UpSampling2	(None, 40, 80, 32)	0
Deconv4 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	4624
Deconv5 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	2320
Deconv6 (Conv2DTranspose with Dropout)	(None, 40, 80, 16)	2320
up_sampling2d_3 (UpSampling2)	(None, 80, 160, 16)	0
Deconv7 (Conv2DTranspose)	(None, 80, 160, 16)	2320
Final (Conv2DTranspose)	(None, 80, 160, 1)	145
Total Number of parameters:		248,685

Table 4.5: Full structure of the model

Two versions of the model were trained, using the same architecture but different dataset:

- ERFNet reduced model: The first model, the ERFNet Reduced model was trained witt a new Dataset, which consisted of the sum of the Baseline Dataset and the FUSO dataset. This training dataset has a total of 13268 images and labels, with a image quality of 160 by 80 pixels. The training parameters used to train the model were the following: batch_size = 128, epochs = 25, pool_size = (2, 2).

- TuSimple ERFNet Reduced model: The second model trained used the TuSImple

dataset, already explained in Chapter 3. This model used the new ERFNet reduced architecture.

The overall performance of the TuSimple ERFNet Reduced model was way superior to the TuSimple Baseline model. However, this new model still carried the same issues the baseline model had. The lane detection was better than the TuSimple Baseline model, being more consistent and in overall, having a better performance in all situations, but still detected several lines besides the EgoLanes during the Rough Road scenarios.

ERFNet TuSimple Model output

(a) Input (b) Merged Output

Figure 4.41: ERFNet_reduced Model Output for Smooth Road

(a) Input (b) Merged Output

Figure 4.42: ERFNet_reduced Model Output at Clear Night for Rough Road

(a) Input

(b) Merged Output

Figure 4.43: ERFNet_TuSimple Model Output at Rain Night for Rough Road

(a) Input

(b) Merged Output

Figure 4.44: ERFNet_TuSimple Model Output at Sunrise .Fog for Rough Road

ERFNet Reduced Model outputs

Day Footage

(a) Input (b) Merged Output

Figure 4.45: ERFNet_reduced Model Output for Smooth Road

(a) Input (b) Merged Output

Figure 4.46: ERFNet_reduced Model Output for Smooth Road with no mark lines

(a) Input (b) Merged Output

Figure 4.47: ERFNet_reduced Model Output for Rough Road

(a) Input (b) Merged Output

Figure 4.48: ERFNet_reduced Model Output in Sharp turn

Rain Day

(a) Input (b) Merged Output

Figure 4.49: ERFNet_reduced Model Output in Rain Day for Smooth Road

(a) Input (b) Merged Output

Figure 4.50: ERFNet_reduced Model Output in Rain Day for Rough Road

Fog Day

(a) Input (b) Merged Output

Figure 4.51: ERFNet_reduced Model Output in Fog for Rough Road

Sunrise

(a) Input (b) Merged Output

Figure 4.52: ERFNet_reduced Model Output in Sunrise Smooth Road

(a) Input (b) Merged Output

Figure 4.53: ERFNet_reduced Model Output in Sunrise Rough Road

Night Footage

(a) Input (b) Merged Output

Figure 4.54: ERFNet_reduced Model Output in Clear Night for Rough

(a) Input (b) Merged Output

Figure 4.55: ERFNet_reduced Model Output for Rain Night Rough Road

(a) Input (b) Merged Output

Figure 4.56: ERFNet_reduced Model Output for Fog Night Rough Road

As all figures show, the ERFNet Reduced model is able to detect the road in all possible situations, with all different types of lightnings and the output prediction does not depend on the Type of road at all. The model is able to detect the road even in the worst possible scenario, which is a Rainy Night detection during the Rough Road part of the circuit.

Chapter 5

Results and Model Comparison

This chapter will include the models comparison and the selection of the best model from to be implemented in the Autonomous driving Testing Truck .

The result of the different models will be compared using two methodologies. The first part will be a visual comparison of the output of all models compared to the baseline model. In the coarse level, the model is expected to predict the Whole road or the two Egolanes, depending on the model, correctly. Two detection errors should be avoided in the processing of lane detection. The first one is missing detection, which predicts the true lane objects in the image as the background, and the second is excessive detection, which wrongly predicts other objects in the background as the lanes. Both these two detection errors will cause the inconsistency of the width of the lane and the number of lanes detected by the EgoLanes models.

The second validation technique will consist in a quantitative calculation based on several metrics, tested with the Test dataset, already explained in Chapter 2. This quantitative analysis will serve as a base to determine effectively which model has the best performance during all different scenarios of the circuit and will give a better way than just a visual recognition of the different outputs to determine the best model performance.

5.1 Models Qualitative Comparison

The following section consist in the comparison and visual analysis of the different models tried during the project.

The analysis is separated into two subsections. The first subsection contains the comparison of the outputs from the Whole Road models, and the second subsection contains the comparison of the EgoLanes models outputs.

To make the different comparisons between models, the SegNet Base model will be used as the Baseline model. The reason to compare all models with this model is because several reasons. First of all, the SegNet Base model was the first model created and trained

with both the Baseline Dataset and the TuSimple Dataset. Other reasons to use this model as the Baseline model is that all the different decisions made regarding the changes to the models were based having the SegNet Base model as a reference.

To establish a better way of comparing the different models, all the input images used to compare the output performance will be the same for all models. The images used are a combination of the different weather conditions and the road type during the circuit. The total images used for these comparison were 48, all part of the Test dataset.

For a visual perception of the model outputs only 4 different scenarios are presented in this memory.

The four images that will be used to show the models output are a Clear Day Smooth road scenario, Clear Day Rough road scenario, Fog Day Rough road scenario and Clear Night Rough road scenario.

5.1.1 Whole Road Detection Models

The Whole Road models have as desired output a prediction of the entire area of the road. This prediction can be determined as optimal if all the area covered by the pixel map fits exactly the shape of the road, without missing areas predicted and without excessive prediction outside the road.

As seen in Chapter 4 section 4.1, the output of the SegNet baseline model has several problems predicting a good pixel-map in some scenarios.

The first model comparison analyzed is between the Retrained model and the Baseline model.

SegNet Base Model vs SegNet Retrained Model

The following images shows the comparison between the SegNet Base model vs SegNet Retrained Model, tested with the same Input images:

(a) Baseline output (b) Retrained Output

Figure 5.1: Smooth Road at Clear Day output prediction comparison

The Figure 5.1 shows the difference between the Baseline model and the Retrained model in the optimal scenario, which is Clear Day and Smooth road. Both models are capable to detect the limits of the road, and therefore not making any excessive prediction. It is possible to see that the Baseline model has some issues to completely detect the whole road, having some empty spaces detected as not as part of the road. The Retrained model is able to give a robust pixel map and the whole road can be inferred from this model.

(a) Baseline output (b) Retrained Output

Figure 5.2: Rough Road at Clear Day output prediction comparison

The first change in the performance of the models can be observed the moment the road type changes from Smooth Road to Rough road. Since the Baseline model has been trained using images of only asphalt road, the output pixel map obtained during the Rough Road is always not good. The Baseline model fails on detecting the limits of the toad, most important part of the detection, and only has as an output some pixel clusters inside the road. The Retrained model, as seen in the Figure 5.2, is able to project a a pixel map covering the whole road, with not excessive prediction and covering the Rough Road in all the length of the road.

The biggest changes between the Baseline model and retrained model can be seen in the Night footage.

(a) Baseline output (b) Retrained Output

Figure 5.3: Rough Road at Clear Night prediction comparison

While the Baseline model do not detect at all the road, only giving some pixel cluster in some frames tested, the Retrained model is capable to give a dense pixel map. The Retrained model performs better than the Baseline model at night. The pixel map is more consistent and has less empty spaces inside the prediction.

However, in Night Scenarios is when the limitations of the Retrained model start to be visible.In the Rough road scenario, the prediction of the Retrained model completely outperforms the Baseline model. In the figure 5.3 it is easy to see that, while the baseline model is not able to detect the road at all, the Retrained model is capable to create a decent pixel map as an output.

The last image showed is from the Fog Scenario.

(a) Baseline output (b) Retrained Output

Figure 5.4: Rough Road with Fog prediction comparison

In this situation is possible to appreciate how, as seen in some night footage, the retrained model has excessive prediction of the lanes.

Although the Retrained model gives a better output performance than the SegNet Base model, the appearance of empty spaces inside the pixel-map where it should have detected

the road and the presence of excessive prediction in repeatedly frames during the video recording, specially at night and in fog scenarios (Figure 5.4). These false positives makes the pixel map not being able to be used to create and detect the road lines.

ERFNet Reduced Model vs Retrained Baseline Model

Even though the comparison between the Baseline model and the ERFNet Model would give some insights between the different architectures, since the ERFNet Reduced model was trained using some FUSO footage, it was decided to make the comparison between the Retrained Baseline Model and the ERFNet Reduced Model, since both models had been trained with Rough Road footage.

The following images shows the comparison between Retrained Baseline Model and the ERFNet Reduced Model, tested with the same Input images:

(a) Baseline output (b) Retrained Output

Figure 5.5: Smooth Road in Clear Day prediction comparison

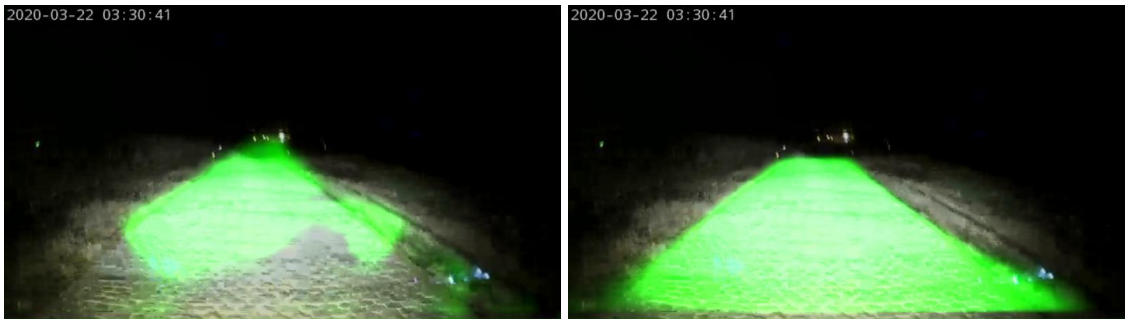
The superior performance of the ERFNet model can be observed even in the Optimal situation. The segmentation pixel map given as output by the ERFNet model is almost perfect, having no missing spaces inside the road and following the margins of the road almost perfect.

(a) Baseline output (b) Retrained Output

Figure 5.6: Rough Road in Clear Day prediction comparison

For the Rough road scenario, it is possible to observe that, while the Retrained model has some missing spots o ¡n the road, the ERFNEt model pixel map covers the whole road, outperforming the Retrained model.

It is possible to observe that the output pixel map is not perfect, having some excessive prediction at the most far away part of the road.



(a) Baseline output (b) Retrained Output

Figure 5.7: Rough Road in Clear Night prediction comparison

It is in the Night scenarios where the can really be observed. Contrary to the SegNet models, both baseline and Retrained, which have issues detecting the margins and giving a dense pixel map, the ERFNet model output is almost the same in Night scenarios compared to Day scenarios. The output even during raining nights was robust enough to detect the road in all frames, having some excessive prediction in just a few of them.

(a) Baseline output (b) Retrained Output

Figure 5.8: Rough Road with Fog prediction comparison

Even though the appearance of some excessive predictions, the amount and the size of them have not a big impact in the final output. As Figure 5.8 shows, the Retrained Baseline model excessive output is more than noticeable in some situations, while the ERFNet model false positives just detects roughly half meter of the sides of the road.

The ERFNet Reduced model only performs poorly in the Sharp turns. Even thought the detection of the road has a better performance than any other model tried before. Compared to the rest of the models in all scenarios the ERFNet model outperforms visibly the other models based on the SegNet architecture, even the Retrained model, which had good output performance during Day scenarios even in Rough Road, but under-performs at night situation.

5.1.2 EgoLane Detection Models

The EgoLanes models have as desired output only the two side lines of the road. The prediction of these lines is more challenging than the prediction of the Whole Road. The models have to be mode precise and the pixel map output have less pixels to form the binary segmentation required.

The output prediction can be determined as optimal y there is only two lanes detected, the two Egolanes, and if the pixel-map is fixed to the margins of the road, without detecting the inside of the road as lanes and covering the entire length of the lines.

The EGolanes model comparison is only analyzed between the SegNet TuSimple Model and the ERFNet TuSimple model. These two models are the only models created using the TuSimple dataset.

The following images shows the comparison of the models tested in with the same Input images

The following images show the comparison between the TuSimple Baseline output and the ERFNet TuSimple output:

(a) SegNet TuSimple Output (b) ERFNet TuSimple Output

Figure 5.9: Smooth Road in Clear Day prediction comparison

In Figure 5.9 it is possible to appreciate that both models are able to detect the EgoLanes. The Segnet TuSimple model detects accurate enough the left line but has some problems to detect the left line in all its length. This erratic detection of the right lane can be seen in all the different outputs gathered during the Testing phase. However, the ERFNet TuSimple model is able to accurately detect both lanes. The good performance of the models in this scenario is caused by the marked lines of the road. Since the training data from TuSimple comes from highway footage, the model is able to give a good prediction when the marked lines are available in the circuit.

(a) SegNet TuSimple Output (b) ERFNet TuSimple Output

Figure 5.10: Rough Road in Clear Day prediction comparison

As seen in Figure 5.10, once the marked lines disappear, the output performance of the TuSimple models goes down immensely. While the SegNet TuSimple model cant barely detect the EgoLanes and generate several lines beside the EgoLanes, the ERFNet TuSimple model is capable to predict only the two egolanes. However, even though the left lines is predicted accurately during the different frames in the Rough Road, the prediction of

the Right Lane is not good and completely shifted. Both models under perform in Rough Road situations and the performance is worse in the Night scenarios, as seen in Figure 5.11.

(a) SegNet TuSimple Output (b) ERFNet TuSimple Output

Figure 5.11: Rough Road in Clear Night prediction comparison

(a) SegNet TuSimple Output (b) ERFNet TuSimple Output

Figure 5.12: Rough Road with Fog prediction comparison

In Fog situation, when the marked lines appear again the ERFNet model is able to perform accurate enpugh to infer the Egolanes and use them to detect the road. However the SegNet model has the same issues as the Clear Day scenario.

After this visual comparison of the Egolane models it is visible that both the SegNet TuSimple model and the ERFNet TuSimple model under-performs at night scenarios, and when the lines are not marked. The performance of both models in Rough Road is also poor, having big excessive prediction of lines that are not the EgoLanes.

The EgoLanes model do not perform good enough to be take into account to be deployed in the Testing Truck. The ERFNet model , if only used in Smooth Road and marked lines are available could be viable.

5.2 Models Quantitative Comparison

This section will include the comparison of the different metrics obtained during the Evaluation phase of the different models.

The models that are going to be Validated are the following:

For Whole Road models, using the Baseline Dataset, FUSO dataset or the Baseline + FUSO dataset, used the Test Dataset as validation ground truth.

- SegNet Baseline model
- SegNet Retrained Model
- ERFNet Based Model

For EgoLanes models, using the TuSimple dataset, the Test TuSimple dataset will be used to measure the different metrics for the following models:

- SegNet TuSimple Model
- ERFNet TuSimple Model

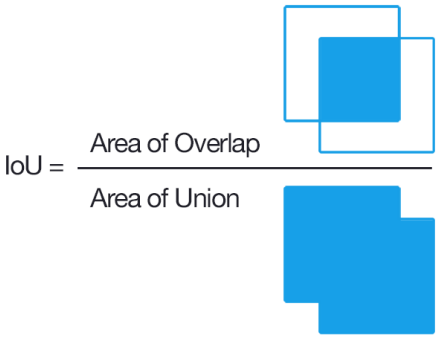
Metrics used to measure model performance

Besides the visual and qualitative testing of all different models, it is necessary to compare their performance in a more quantitative way. To do this quantitative performance test, several measurements are take in count.

Since the Lane Detection problem is a binary segmentation classification problem, the best way to measure any semantic segmentation model is using the Intersection on Union or Jaccard Index and the F1 Score or Dice coefficient. Also, Precision and Recall will be used to have a more overall view of the total accuracy of each model.

- **Intersection on Union or Jaccard Index**: "The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally given the French name coefficient de communauté by Paul Jaccard), is a statistic used for gauging the similarity and diversity of sample sets." [15]. It will measure the overall accuracy of the output predicted over the ground truth label.

  The intersection $(A \cap B)$ is comprised of the pixels found in both the prediction mask and the ground truth mask, whereas the union $(A \cup B)$ is simply comprised of all pixels found in either the prediction or target mask.



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Figure 5.13: Visual representation of IoU and Jaccard equation

- **Precision and Recall**:"In pattern recognition, information retrieval and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are therefore based on an understanding and measure of relevance." [16]. These two measurements will tell the percentage of pixels from the predicted output form part of the ground truth label (Precision) and the percentage of the ground truth label, the road, is detected by the predicted output (Recall).

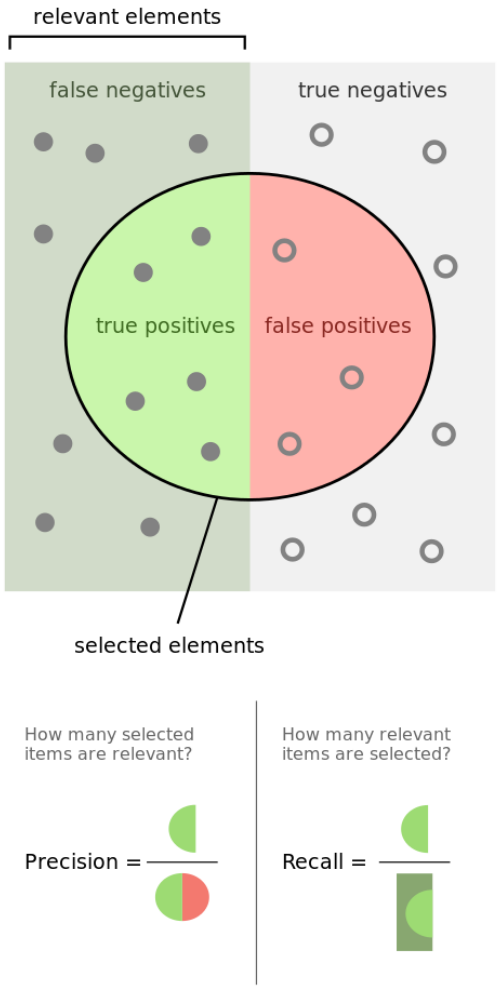


Figure 5.14: Visual representation of Precision and Recall

- **F1 score or Dice**: "The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall). The F1 score is also known as the Sørensen–Dice coefficient or Dice similarity coefficient (DSC)." [17]. This coefficient will take into account both Precision and Recall to calculate the accuracy of the output prediction. The formula to calculate the F_1 is the following:

$$F_1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$

With these four coefficients, all models will be measured using the same Test dataset (Test dataset for Whole Road models, TuSimple Test dataset for Egolane models) in a

quantitative way. After all models output description a table with the values of this coefficients will be listed.

5.2.1 Whole Road Models

SegNet Baseline Model Metrics

The next table shows the results obtained during the validation phase of the SegNet Baseline model:

Baseline Evaluation Results

Model	Precision	Recall	IoU	F1
Baseline Model	0.8359	0.8054	0.7021	0.8146

Table 5.1: Baseline model test metrics

Baseline Retrained Model

Same as the Baseline model, the Retrained_5 model was tested using the Test Dataset to test the Accuracy of the model.

The next table shows the results obtained during the validation phase of the Retrained Baseline model:

Retrained Baseline Evaluation Results

Model	Precision	Recall	IoU	F1
Retrained_5 Model	0.8227	0.9357	0.7798	0.8709

Table 5.2: Retrained_5 model test metrics

ERFNet Based Model

The next table shows the results obtained during the validation phase of the ERFNet Reduced Model model:

ERFNet Evaluation Results

Model	Precision	Recall	IoU	F1
ERFNet_reduced Model	0.9026	0.9603	0.8703	0.9283

Table 5.3: ERFNet Reduced model test metrics

5.2.2 EgoLanes Models

SegNet Based TuSimple Model

The next table shows the results obtained during the validation phase of the SegNet TuSimple model:

TuSimple Baseline Evaluation Results

Model	Precision	Recall	IoU	F1
TuSimple Base Model	0.4705	0.2361	0.1875	0.3071

Table 5.4: Baseline model test metrics

ERFNet Based TuSimple Model

The next table shows the results obtained during the validation phase of the ERFNet TuSimple model:

ERFNet TuSImple Evaluation Results

Model	Precision	Recall	IoU	F1
ERFNet_TuSimple Model	0.5137	0.5030	0.3528	0.5015

Table 5.5: ERFNet TuSimple model test metrics

5.2.3 Metrics comparison

After validate all models and computed all models metrics results a comparison of all the different metrics can be seen in the following table:

Models Metrics Evaluation Results

Model	Precision	Recall	IoU	F1
Baseline Model	0.8359	0.8054	0.7021	0.8146
Retrained_5 Model	0.8227	0.9357	0.7798	0.8709
ERFNet_reduced Model	**0.9026**	**0.9603**	**0.8703**	**0.9283**
TuSimple Base Model	0.4705	0.2361	0.1875	0.3071
ERFNet_TuSimple Model	0.5137	0.5030	0.3528	0.5015

Table 5.6: ERFNet TuSimple model test metrics

The results obtained shows a big difference between the Whole Road models and the EgoLane models. This main difference can be caused by the different dataset used for each type of model. Also, the Egolanes models had a lower number of pixels in each ground truth label to compare, plus both Egolanes model had several excessive prediction of lanes which were not part of the Egolanes. These two reasons made the values of the metrics of the Egolanes model to drop below 0.5 in all metrics. Even though the low results, it is visible the big increase in the Recall of the ERFNet TuSimple Based model vs the SegNet TuSimple model.

Due to these low results,the EgoLanes model will not being taken in consideration to be chosen as the Final model to be deployed in the FUSO Autonomous Testing Truck in the Kitsuregawa testing Facility.

On regard of the Whole Road models, the Baseline model starts with a IoU of 0.7 and an F1 of 0.81. Being the first model developed the results are good. However, these high values of IoU and F1 contrast with what have been seen in the visual comparison and the visual representation of the output. The main reason of these values can be because the size of the label ground truth compared to the total size of the image. This will increment the values of the metrics even though the models do not perform in a very high accuracy.

Despite this bias on the metrics, it is possible to appreciate the increase of the values of all metrics in both the Retrained model and the ERFNet model compared to the Baseline model.

The Retrained model has a slightly lower Precision compared to the Baseline model, which means that it has a little more false positives pixels than the Baseline model, but there is a huge increase in the Recall performance, with 0.13 more than the Baseline models recall. this mean that the Retrained model output has less False Negatives, and the predicted pixel map covers a bigger area of the road despite not being as precise as the Baseline Model. Taking the IoU and the F1 metrics is possible to see that the Retrained model is 7% more accurate than the Baseline model.

From all the models Tested during the development of the project, the ERFNet Reduced model has obtained the best results in all the metrics available. It has a 0.9 Precision and a 0.96 Recall, 0.07 and 0.16 points higher than the Baseline model respectively, and it has a F1 of 0.9283, being the first model to surpas the 0.9 value in this metric. The ERFNet model also has the best overall accuracy of all models, with a 0.8703 IoU, 10% higher than the Retrain model and a 17% higher than the Baseline model.

5.3 Final Results

After the Qualitative analysis and the Quantitative metrics comparison of the different models, it can be observed that the ERFNet model, both using the Quantitative and the

Qualitative Visual methods, is the best performing model in all different Scenarios proposed in the Kitsuregawa Testing Circuit.

Using the visual Qualitative methodology to determine the best model, the ERFNet reduced model has the best performance of all the models tried. The model is able to predict the road in a accurate way and has both good detection and low excessive prediction of the road, except in some night scenarios. It is noticeable to add that the Final model also has a good performance in continuous frames, and the prediction between consecutive frames are very similar.

However, this continuous prediction is not always present. In the Qualitative testing videos it was possible to observe during the Night Scenarios that, due to changes of lightning (for security reasons there is in the Kitsuregawa Testing Circuit a intermittent blue light which interferes with the lights of the truck and changes the lightning of the road), the prediction between some frames changes drastically and leads to creation of a very different pixel map of the road. These effect can be seen in the following images:

(a) First Frame Output (b) Second Frame Output

Figure 5.15: Consecutive frames obtained from the Night Raining Video Testing of the ERFNet model. It is possible to observe the difference between predictions from one frame to the other. In the second Frame, due to the high lightning, the model predicts the security light as part of the road, creating a non accurate pixel map putput.

Despite the some minor problems, such as the sporadic excessive prediction seen in Figure 5.15, the model performs in a robust way to be able to continuously detect the Road in any condition.

Chapter 6

Conclusion and Future Steps

This chapter includes the conclusions regarding the whole process of development of the models and the future steps that may be taken in order to improve the Final model to generate a even more robust and continuous segmentation pixel map of the road.

During the development of the project several aspects were taken in consideration to determine the best possible model.

The election of the SegNet architecture as the baseline of the model was because previous models based on SegNet had performed in good conditions. The main reason the Baseline model did not performed correctly during the Test phase was mainly due to the difference between the Test road and the training images road. This difference was more clear in the Rough Road scenarios and and in night situations. One possible option that was presented during the development of the project was to expand the Neural Network, adding some layers at the beginning and at the end of the Network and train them with the Circuit images. The main issue observed was that the it was not possible to transfer the weights of the different Convolutional layers from the SegNet Baseline model, making that approach not viable and at the end opted for the Transfer Learning and retrain the Baseline model with some inside FUSO footage.

Since the tune fitting helped to improve the output performance, it was clear that introducing as training images footage from the Kitsuregawa testing facility circuit helped to improve the model, it was decided to train the next iteration of models with both the Baseline dataset and the FUSO dataset.

Even though the improvement on the performance of the model after the introduction of retraining the model with FUSO data, it was noticeable that the model was still not able to gather all the different factions of the road and it was visible that some information were being lost in the network. One possible reason of this was the inner layers of the model. The center of the SegNet architecture was a final Max Pooling followed for a Upsampling layer. It was possible that in that dimensional reduction of the outputs of the inner 2D Convolutional layers some valuable information of the road was missing.

This was the reason to create the new architecture based on the ERFNet layer struc-

ture, the ERFNet Reduced architecture. The main difference was the new architecture did not have a mirrored Encoder-Decoder structure, and avoided the last centered Max Pooling Upsampling layers. Also, adding more inner layers inside the architecture helped to generate a more robust output throughout all different scenarios

Regarding on how to test this kind of semantic segmentation models, the final results of the metrics are strongly defined by the real size of the pixel map that wants to be predicted. The enormous gap between the values obtained for the EgoLanes models and the Whole Road models were too big that the models could not be compared between them. The area of the ground truth labels of the Whole Road models covers almost a third of the image, making easier to have a better percentage of predicted pixels inside the testing label. On the other hand, the EgoLanes models ground truth testing labels only covers a really small amount of the image, and making the values of the metrics to plummet compared to the Whole Road models. Therefore, the testing method needs to be improved to be able to compare the accuracy of both types of models.

In conclusion, the Final model deployed at the Autonomous Testing Truck is capable to detect the Road of the Testing Circuit in all different scenarios proposed at the beginning of the project, facing some small issues at Night scenarios and some intermittent problems caused by the non-stability of the input camera.

Future Steps

Nevertheless, there is space to improve the model and also to increment the possibilities of it.

Some possible steps to make improvements in the model and continue with the project are exposed here:

- Improve the output performance in sharp turns. The actual model has problems to generate a decent predictions during the sharp turns of the circuit. As exposed before, the turn is extremely close, making the road to only appear at the bottom of the image and losing all similarities with the rest of the circuit. These can be prevented by training the model with more sharp turns footage.

- Increase the number of classes to detect different object apart from the road. One improvement that can be implemented in the model is to train it not be able to detect other parts of the circuit, such as fences, trees and bushes. Being able to detect the fences and bushes will help to prevent some excessive prediction that occurs to the actual model during some scenarios. It will also help to take the model to next level and not only use the Road Lane Detection model to detect possible deviation, but to use it to stop in case of obstacles in the road.

- Introduction of LSTM network. During the first stages of the development of the model, one idea proposed was to implement and embedded LSTM Convolutional

network right after the Encoder structure. This idea was based on the Robust Lane Detection from Continuous Driving Scenes paper [9] [18] .The idea behind this was to, instead predict the road using only one frame at a time, generate the prediction taking into account the previous generated predictions, saving the output of the last inner 2D Convolutional layer for a specific number of frames as a time series.

- Generate more data for the EgoLanes models. One main reason why the EgoLanes models did not perform good enough was because all the footage used to train the Neural Networks were taken always from the highway, with marked lines. Increasing the existing dataset with FUSO footage including the same type of labeling as the TuSimple dataset could improve the performance of the models based on detecting the EgoLanes.

Appendix A

This Annex includes the output of a selected frames from the Test Dataset and the output of each frame. The total number of frames showed are 20, taken from all different scenarios that exist in the Test Dataset.

- Frames 1 to 4 are from Clear Night scenario.
- Frames 5 to 9 are from Rain Night scenario
- Frames from 10 to 13 are from Clear Day scenario.
- Frames from 14 to 17 are fro Fog Day scenario.
- Frames from 18 to 20 are from Rain Day scenario.

A.1 SegNet Base model outputs

(a) Input Frame (b) Output

Figure A.1: Segnet Base output 1

(a) Input Frame (b) Output

Figure A.2: Segnet Base output 1

(a) Input Frame (b) Output

Figure A.3: Segnet Base output 2

v

(a) Input Frame (b) Output

Figure A.4: Segnet Base output 3

(a) Input Frame (b) Output

Figure A.5: Segnet Base output 4

(a) Input Frame (b) Output

Figure A.6: Segnet Base output 5

(a) Input Frame (b) Output

Figure A.7: Segnet Base output 6

(a) Input Frame (b) Output

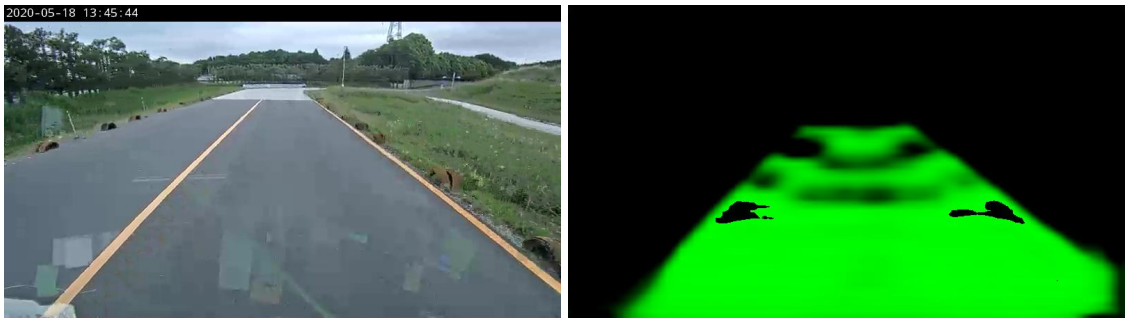
Figure A.8: Segnet Base output 7

(a) Input Frame (b) Output

Figure A.9: Segnet Base output 8

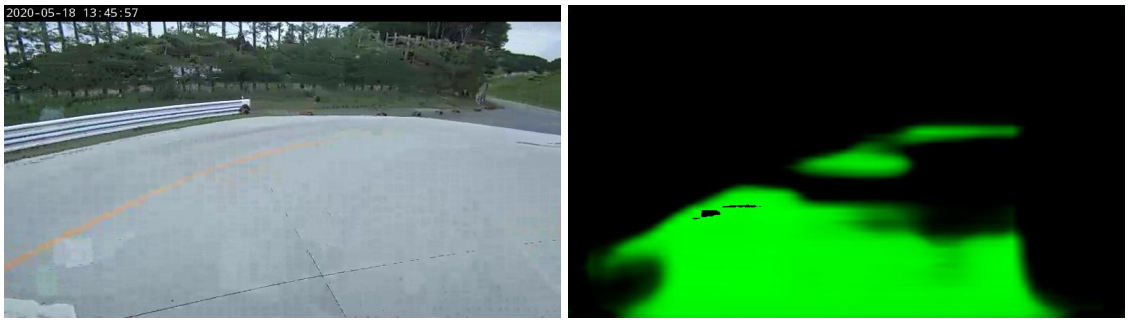
(a) Input Frame (b) Output

Figure A.10: Segnet Base output 9



(a) Input Frame (b) Output

Figure A.11: Segnet Base output 10



(a) Input Frame (b) Output

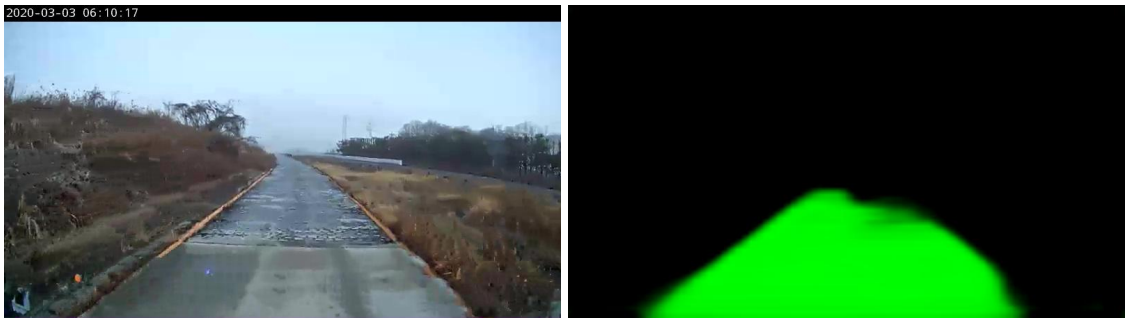
Figure A.12: Segnet Base output 11

(a) Input Frame (b) Output

Figure A.13: Segnet Base output 12

(a) Input Frame (b) Output

Figure A.14: Segnet Base output 13



(a) Input Frame (b) Output

Figure A.15: Segnet Base output 14

(a) Input Frame (b) Output

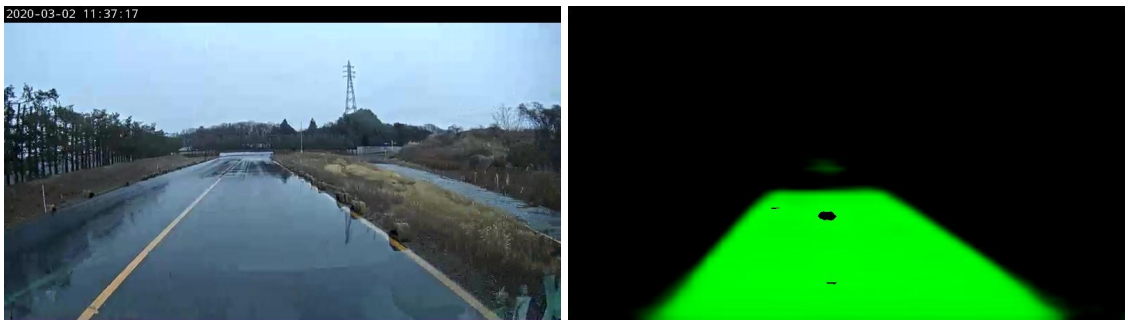
Figure A.16: Segnet Base output 15

(a) Input Frame (b) Output

Figure A.17: Segnet Base output 16

(a) Input Frame (b) Output

Figure A.18: Segnet Base output 17



(a) Input Frame (b) Output

Figure A.19: Segnet Base output 18

(a) Input Frame (b) Output

Figure A.20: Segnet Base output 19

(a) Input Frame (b) Output

Figure A.21: Segnet Base output 20

A.2 Retrained model outputs

(a) Input Frame (b) Output

Figure A.22: Retrained output 1

(a) Input Frame (b) Output

Figure A.23: Retrained output 2

(a) Input Frame (b) Output

Figure A.24: Retrained output 3

(a) Input Frame (b) Output

Figure A.25: Retrained output 4

(a) Input Frame (b) Output

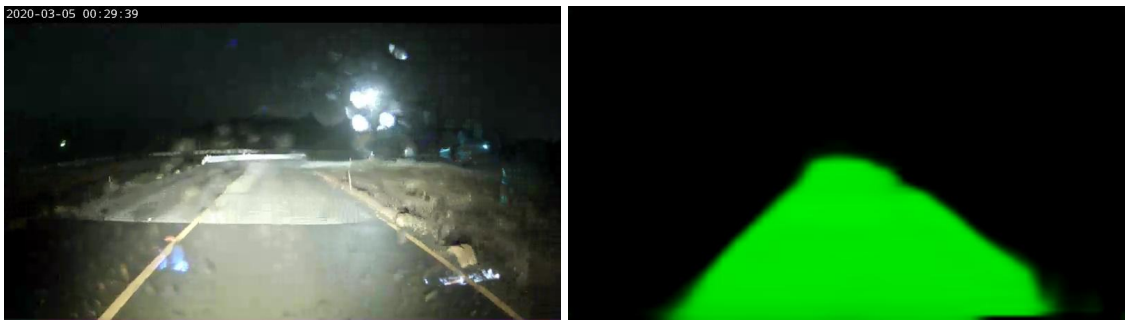
Figure A.26: Retrained output 5

(a) Input Frame (b) Output

Figure A.27: Retrained output 6

(a) Input Frame (b) Output

Figure A.28: Retrained output 7



(a) Input Frame (b) Output

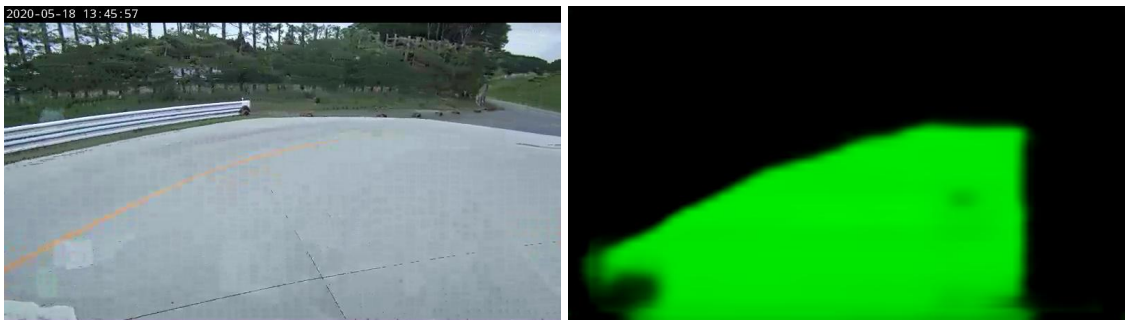
Figure A.29: Retrained output 8

(a) Input Frame (b) Output

Figure A.30: Retrained output 9

(a) Input Frame (b) Output

Figure A.31: Retrained output 10

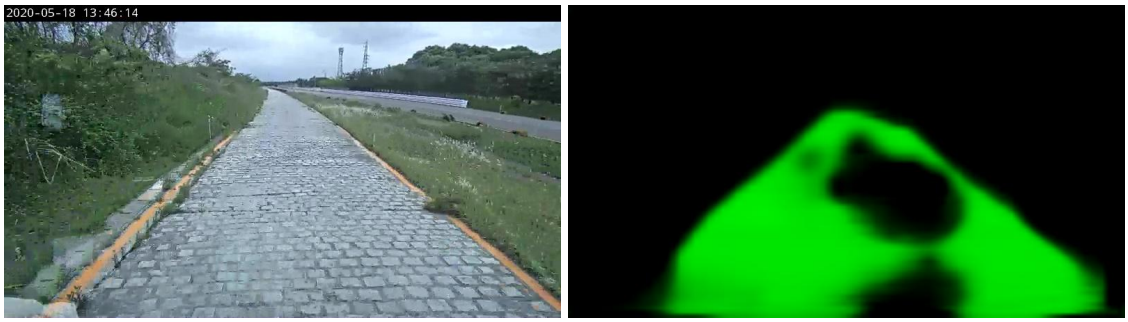


(a) Input Frame (b) Output

Figure A.32: Retrained output 11

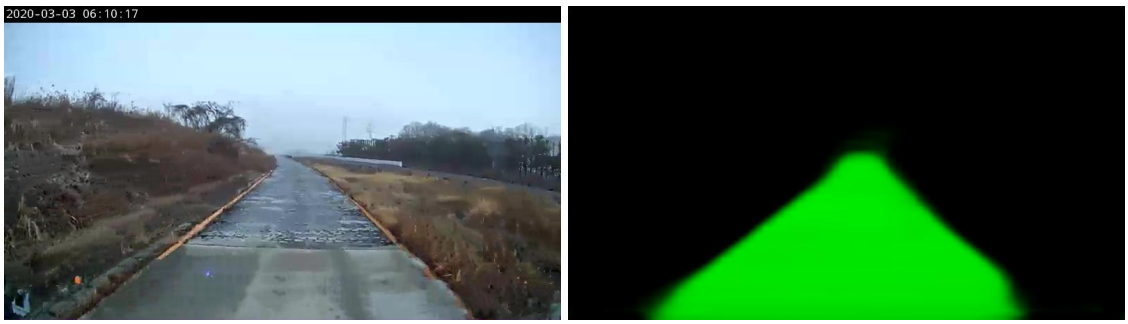
(a) Input Frame (b) Output

Figure A.33: Retrained output 12



(a) Input Frame (b) Output

Figure A.34: Retrained output 13



(a) Input Frame (b) Output

Figure A.35: Retrained output 14

(a) Input Frame (b) Output

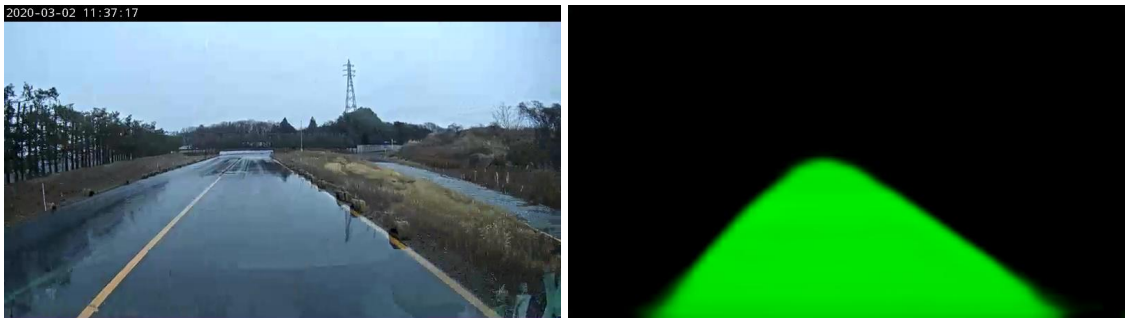
Figure A.36: Retrained output 15

(a) Input Frame (b) Output

Figure A.37: Retrained output 16

(a) Input Frame (b) Output

Figure A.38: Retrained output 17



(a) Input Frame (b) Output

Figure A.39: Retrained output 18

(a) Input Frame (b) Output

Figure A.40: Retrained output 19

(a) Input Frame (b) Output

Figure A.41: Retrained output 20

A.3 ERFNet reduced model

(a) Input Frame (b) Output

Figure A.42: ERFNet reduced output 1

(a) Input Frame (b) Output

Figure A.43: ERFNet reduced output 2

(a) Input Frame (b) Output

Figure A.44: ERFNet reduced output 3

(a) Input Frame (b) Output

Figure A.45: ERFNet reduced output 4

(a) Input Frame (b) Output

Figure A.46: ERFNet reduced output 5

(a) Input Frame (b) Output

Figure A.47: ERFNet reduced output 6

(a) Input Frame (b) Output

Figure A.48: ERFNet reduced output 7

(a) Input Frame (b) Output

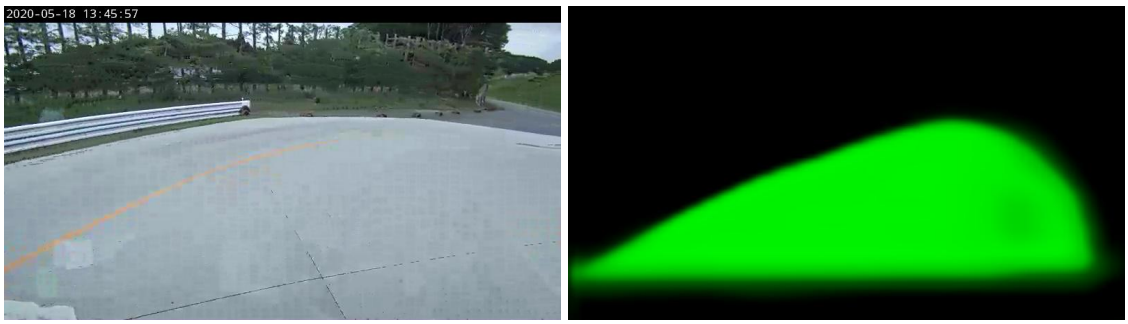
Figure A.49: ERFNet reduced output 8

(a) Input Frame (b) Output

Figure A.50: ERFNet reduced output 9

(a) Input Frame (b) Output

Figure A.51: ERFNet reduced output 10



(a) Input Frame (b) Output

Figure A.52: ERFNet reduced output 11

(a) Input Frame (b) Output

Figure A.53: ERFNet reduced output 12

(a) Input Frame (b) Output

Figure A.54: ERFNet reduced output 13

(a) Input Frame (b) Output

Figure A.55: ERFNet reduced output 14

(a) Input Frame (b) Output

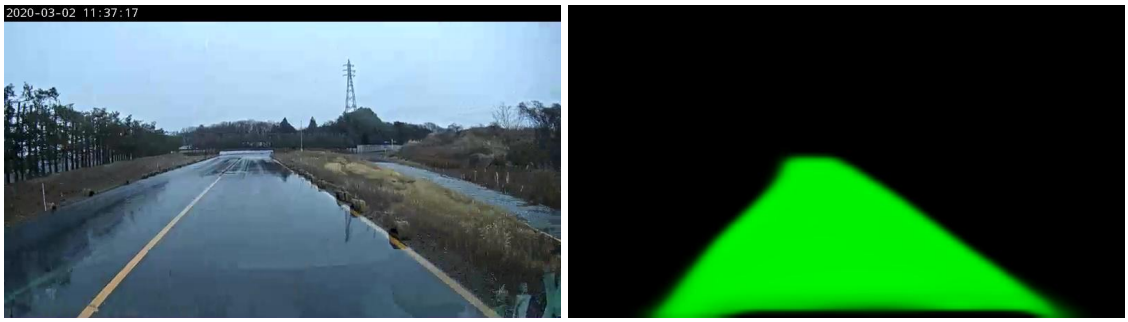
Figure A.56: ERFNet reduced output 15

(a) Input Frame (b) Output

Figure A.57: ERFNet reduced output 16

(a) Input Frame (b) Output

Figure A.58: ERFNet reduced output 17



(a) Input Frame (b) Output

Figure A.59: ERFNet reduced output 18

(a) Input Frame (b) Output

Figure A.60: ERFNet reduced output 19

(a) Input Frame (b) Output

Figure A.61: ERFNet reduced output 20

A.4 SegNet TuSimple model

(a) Input Frame (b) Output

Figure A.62: SegNet TuSimple output 1

(a) Input Frame (b) Output

Figure A.63: SegNet TuSimple output 2

(a) Input Frame (b) Output

Figure A.64: SegNet TuSimple output 3

(a) Input Frame (b) Output

Figure A.65: SegNet TuSimple output 4

(a) Input Frame (b) Output

Figure A.66: SegNet TuSimple output 5

(a) Input Frame (b) Output

Figure A.67: SegNet TuSimple output 6

(a) Input Frame (b) Output

Figure A.68: SegNet TuSimple output 7

(a) Input Frame (b) Output

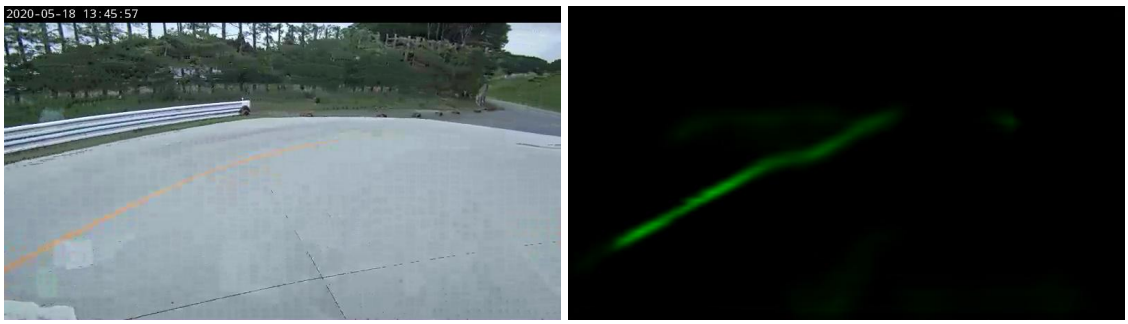
Figure A.69: SegNet TuSimple output 8

(a) Input Frame (b) Output

Figure A.70: SegNet TuSimple output 9

(a) Input Frame (b) Output

Figure A.71: SegNet TuSimple output 10



(a) Input Frame (b) Output

Figure A.72: SegNet TuSimple output 11

(a) Input Frame (b) Output

Figure A.73: SegNet TuSimple output 12

(a) Input Frame (b) Output

Figure A.74: SegNet TuSimple output 13

(a) Input Frame (b) Output

Figure A.75: SegNet TuSimple output 14

(a) Input Frame (b) Output

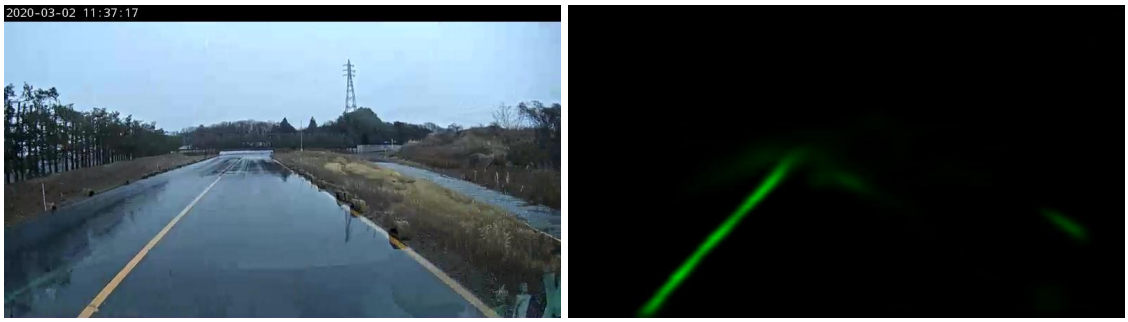
Figure A.76: SegNet TuSimple output 15

(a) Input Frame (b) Output

Figure A.77: SegNet TuSimple output 16

(a) Input Frame (b) Output

Figure A.78: SegNet TuSimple output 17



(a) Input Frame (b) Output

Figure A.79: SegNet TuSimple output 18

(a) Input Frame (b) Output

Figure A.80: SegNet TuSimple output 19

(a) Input Frame (b) Output

Figure A.81: SegNet TuSimple output 20

A.5 ERFNet TuSImple model

(a) Input Frame (b) Output

Figure A.82: ERFNet TuSimple output 1

(a) Input Frame (b) Output

Figure A.83: ERFNet TuSimple output 2

(a) Input Frame (b) Output

Figure A.84: ERFNet TuSimple output 3

(a) Input Frame (b) Output

Figure A.85: ERFNet TuSimple output 4

(a) Input Frame (b) Output

Figure A.86: ERFNet TuSimple output 5

(a) Input Frame (b) Output

Figure A.87: ERFNet TuSimple output 6

(a) Input Frame (b) Output

Figure A.88: ERFNet TuSimple output 7

(a) Input Frame (b) Output

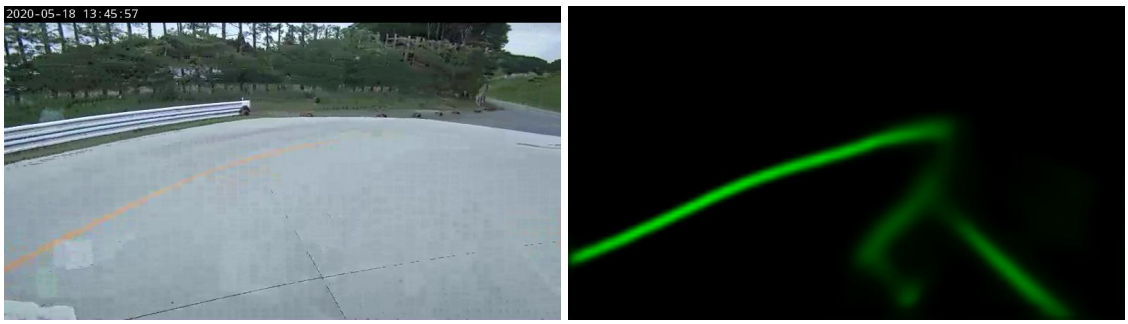
Figure A.89: ERFNet TuSimple output 8

(a) Input Frame (b) Output

Figure A.90: ERFNet TuSimple output 9

(a) Input Frame (b) Output

Figure A.91: ERFNet TuSimple output 10



(a) Input Frame (b) Output

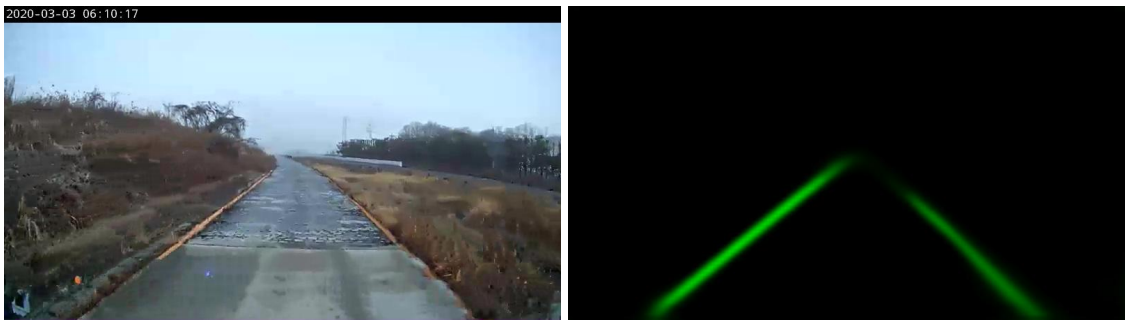
Figure A.92: ERFNet TuSimple output 11

(a) Input Frame (b) Output

Figure A.93: ERFNet TuSimple output 12

(a) Input Frame (b) Output

Figure A.94: ERFNet TuSimple output 13



(a) Input Frame (b) Output

Figure A.95: ERFNet TuSimple output 14

(a) Input Frame (b) Output

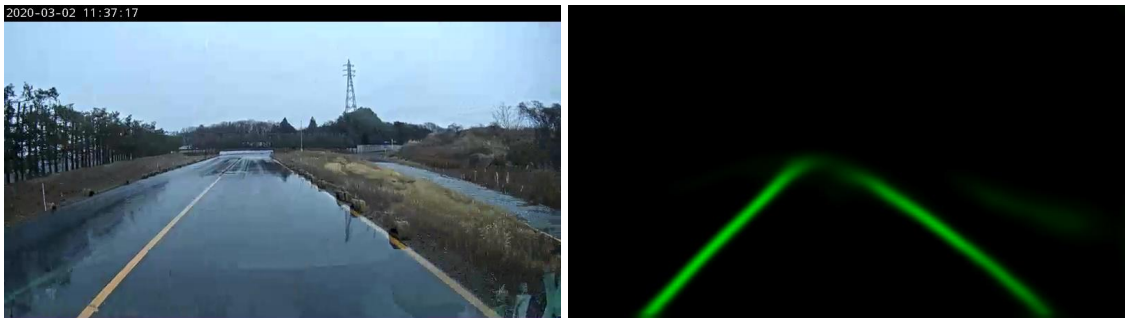
Figure A.96: ERFNet TuSimple output 15

(a) Input Frame (b) Output

Figure A.97: ERFNet TuSimple output 16

(a) Input Frame (b) Output

Figure A.98: ERFNet TuSimple output 17



(a) Input Frame (b) Output

Figure A.99: ERFNet TuSimple output 18

(a) Input Frame (b) Output

Figure A.100: ERFNet TuSimple output 19

(a) Input Frame (b) Output

Figure A.101: ERFNet TuSimple output 20

Appendix B

B.1 Sustainable Development Goals

The Sustainable Development Goals (SDGs) are a collection of 17 global goals designed to be a "blueprint to achieve a better and more sustainable future for all"

These goals will help on the development of a more sustainable world and society, making progress in reducing the differences between people by attacking the main issues of the world such as poverty, hunger, education and sustainability of the ecosystem.

The final goal of this project is to reduce the accidents of the vehicles while testing their durability. This testing is required to be applied under hazardous conditions, which will may put on risk the health and lives of the workers involved in the testing.

The main Sustainable Development Goals focused on this project are number 3, Good Health and number 9, Industry, Innovation and Infrastructure.

For the SDG 3, the lane detection project will try to avoid unnecessary vehicle accidents. The main goal of autonomous driving is to help the drivers to avoid accidents, which will lead to saving lives. The autonomous driving technologies are helping right now as ADAS systems during the driving. One of the most important parts of these systems is the road detection, which is essential to help the driver during difficult situations and even take charge of the driving if necessary.

Thank to the Lane Detection model, the vehicles will be able to detect possible changes of direction and deviations from the road and notify them to the driver, and, in case of emergency, stop the vehicle and avoid an accident.

With the application of this project, it will be possible to avoid around 30 to 40 accidents per year during the testing of the different vehicles, and, therefore, avoid possible health issues caused to the drivers in this accidents.

For the SDG 9, it is important to notice the impact and the different applications of this Lane Detection Model. With a robust lane detection it will be possible to implement real self driving vehicles only using a camera to detect different objects of the road. Being able to detect the road continuously, any self driving vehicle will know exactly the path it

must follow and at the end, being able to drive alone without any human intervention. If the technology on this topic keeps evolving, is very possible to have access to self-driving cars and different vehicles in less than 10 years, allowing to impulse some industries, like transportation and aviation, which requires important human resources to be efficient.

Bibliography

[1] Labor Tochigi Prefecture Department of Industry and Tourism. *Guide to Industrial locations in Tochigi*. URL: http://www.pref.tochigi.lg.jp/kogyo/english/voice/031.html.

[2] John Canny. "A Computational Approach To Edge Detection". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (Dec. 1986), pp. 679–698.

[3] Shengyan Zhou et al. "A Novel Lane Detection based on Geometrical Model and Gabor Filter". In: July 2010, pp. 59–64. DOI: 10.1109/IVS.2010.5548087.

[4] Richard O. Duda and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures". In: (1972). URL: https://dl.acm.org/doi/10.1145/361237.361242.

[5] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[6] Vijay Badrinarayan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Vehicular Technology* (2016).

[7] Vijay Badrinarayan, Alex Kendall, and Roberto Cipolla. *SegNet*. URL: https://mi.eng.cam.ac.uk/projects/segnet/.

[8] E. Romera et al. "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation". In: *IEEE Transactions on Intelligent Transportation Systems* 19.1 (2018), pp. 263–272.

[9] Q. Zou et al. "Robust lane detection from continuous driving scenes using deep neural networks". In: *IEEE Transactions on Vehicular Technology* (2019).

[10] Michael Virgo. "Lane Detection with Deep Learning". In: (2017).

[11] Michael Virgo. *Michael Virgo Github*. URL: https://github.com/TuSimple/tusimple-benchmark/issues/3.

[12] TuSimple. *TuSimple Github*. URL: https://github.com/TuSimple/tusimple-benchmark/issues/3.

[13] Keras. *Keras Reference*. URL: https://keras.io/.

[14] OpenCV. *Image Blurring (Image Smoothing)*. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html.

[15] Wikipedia. *Jaccard index*. URL: https://en.wikipedia.org/wiki/Jaccard_index.

[16] Wikipedia. *Precision and Recall*. URL: https://en.wikipedia.org/wiki/Precision_and_recall.

[17] Wikipedia. *F1 score*. URL: https://en.wikipedia.org/wiki/F1_score.

[18] Zou Q et al. *Robust Lane Detection Github*. URL: https://github.com/qinnzou/Robust-Lane-Detection.

[19] François Chollet. *Deep Learning with Python*. 2018. ISBN: 9781617294433.

[20] Arthur Ouaknine. *Review of Deep Learning Algorithms for Image Semantic Segmentation*. URL: https://medium.com/@arthur_ouaknine/review-of-deep-learning-algorithms-for-image-semantic-segmentation-509a600f7b57.

[21] Davy Neven et al. "Towards End-to-End Lane Detection: an Instance Segmentation Approach". In: (2018).

[22] Ze Wang, Weiqiang Ren, and Qiang Qiu. "LaneNet: Real-Time Lane Detection Networks for Autonomous Driving". In: (2018). URL: https://arxiv.org/abs/1807.01726.

[23] Javier Ibanez-Guzman et al. "Autonomous Driving: Context and State-of-the-Art". In: Mar. 2012. ISBN: 978-0-85729-084-7. DOI: 10.1007/978-0-85729-085-4_50.

[24] OpenCV. *Canny Edge Detection*. URL: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html.

[25] Joe Schueller. *Lane Detection Systems*. 2016. URL: http://joe-schueller.github.io/2016/04/03/lane-detection-systems.html.