Driver Monitoring System for Outside World Context

Adrian-Răzvan Iordache

coordinated by

Associate Professor Bogdan Alexe

Faculty of Mathematics and Computer Science
University of Bucharest

July 1, 2022

Table of contents

Introduction

Problem context

Our solution *aims to improve the domain of driver monitoring and assistance for video telematics on embedded devices. Using a Snapdragon processor with a machine learning hardware accelerator integrated, our main focus is the trade-off between model complexity/accuracy and latency.*

Our solution

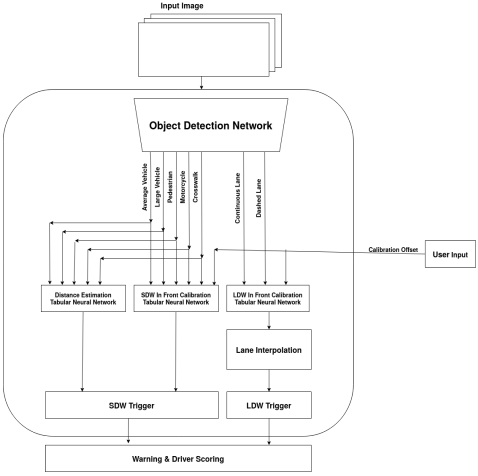
We propose a solution based on a single camera visual input for ***Safe Distance Warning*** *and* ***Lane Departure Warning systems*** *based on multiple stacked deep neural networks for various tasks and a computer vision algorithm for detection interpolation using accumulative masks.*

Architecture of the solution proposed

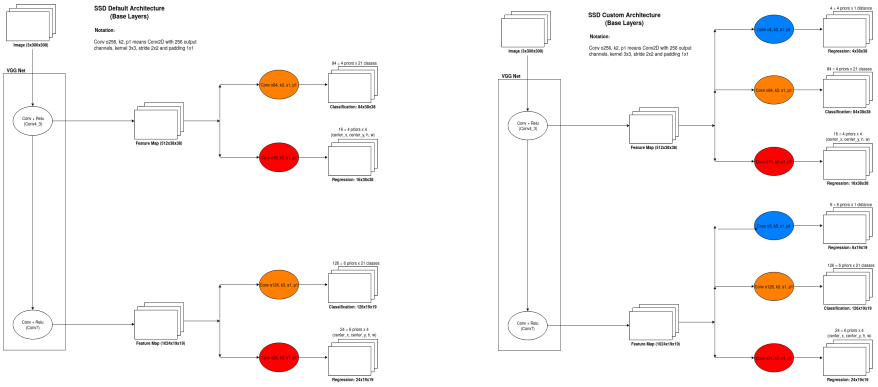
We summarize our ideas in the following lines:

1. A **base network** used for object detection over different classes
2. Three **stacked neural networks** used over the base network for solving various tasks such as:
   - Estimating the distance between our vehicle and the detected objects
   - Detecting the objects in front of the vehicle, invariant to the camera position on the windshield
3. A computer vision interpolation algorithm for predicting bounding boxes used for lane detection

Solution Overview



Estimating distances from a single frame

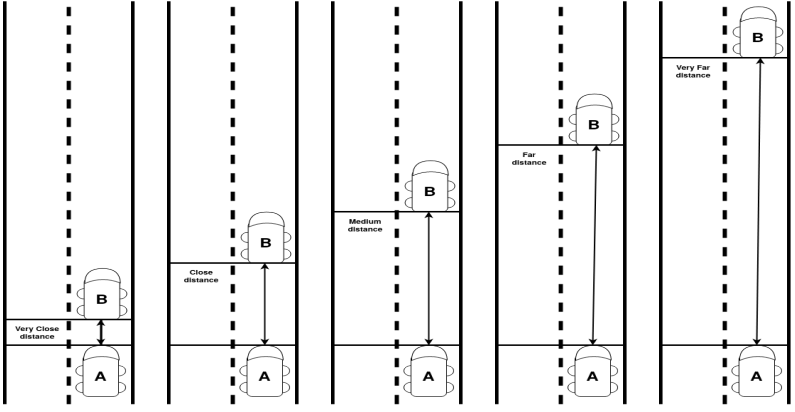


(left) The default architecture of Single Shot Detector with VGG16 for predicting boxes and classes.

(right) A custom architecture of Single Shot Detector with VGG16 for also extracting the estimated distance.

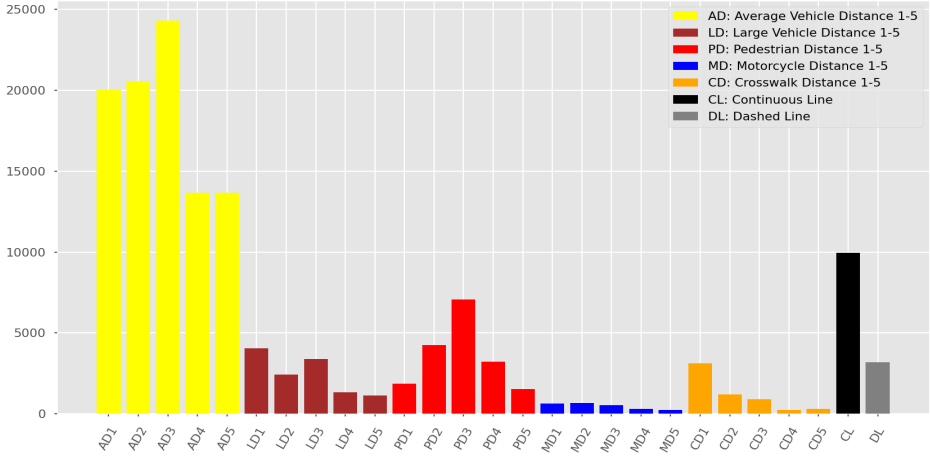
Unfortunately, the second approach proved to be impractical in the field due to latency issues.

Mapping continuous distances in discrete categories

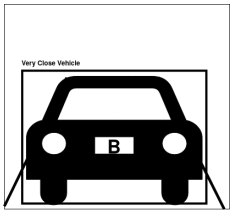


For estimating distance, the solution came from the concept of mapping the continuous space of distances in discrete categories. In the initial stages, this labeling strategy is applied to the default architecture of an object detector. This generates a problem of classification and localization for over 27 object classes.

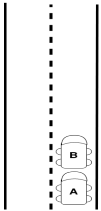
Label distribution for the 27 classes object detection problem



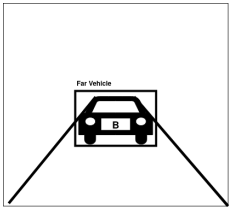
Estimating the distance based on classes and bounding boxes



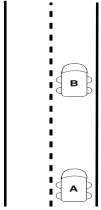
(a)



(b)



(c)



(d)

End-To-End vs. Multi-Network approach

As this End-To-End method proved to be practical in the field, we compare this possible solution with a stacked multi-network approach. The second method came from the intuition that for estimating distances we only need the box coordinates and the predicted class.

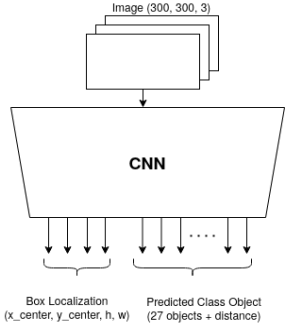
Example Description

(a) Very close vehicle viewed from in front perspective.
(b) The same vehicle from (a) viewed from above.
(c) Far vehicle viewed from in front perspective.
(d) The same vehicle from (c) is viewed from above.

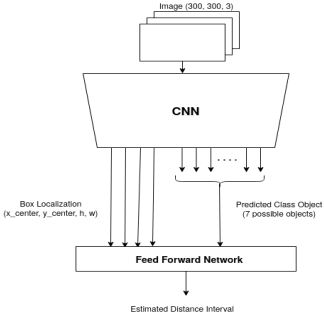
Predicting distances based on bounding boxes

We see how for a specific detected category (average size vehicle), we can train an estimator based on various bounding boxes to estimate the distance. To use only one estimator to predict the distance for all object categories, we need to use as input feature the predicted category, besides the bounding box.

End-To-End Network vs. Multi-Network approach



(a)



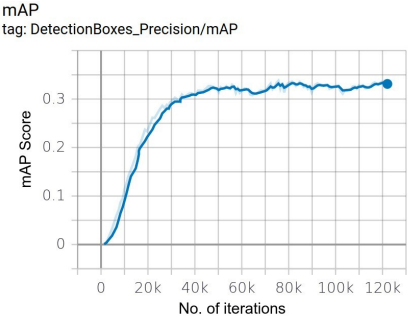
(b)

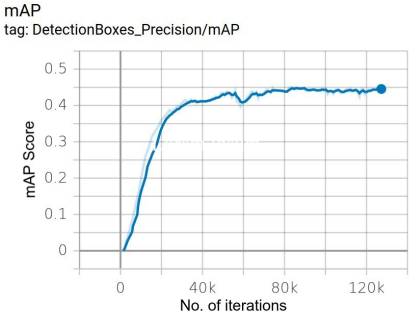
(a) End to end network, 5 of 7 classes have attached the estimated distance (ex. very close ***pedestrian****) resulting in 25 SDW classes and 2 LDW classes. (b) Multi-Network, the CNN predicts the initial 7 classes, (ex.* ***pedestrian****), the distance is predicted by a stacked neural network using the bounding box and the detection class as inputs.*

Validation Results

Method*	Iteration Step	mAP**	mAP@.50IOU	mAP@.75IOU
Multi-Network	83,880	**45.0%**	**69.72%**	**46.33%**
End to End Network	77,250	33.4%	51.08%	34.2%

(*) Only for object detection, based on a quantized Single Shot Detector with a MobileNetV2.
(**) The AP is averaged over 10 Intersection over Union (IoU) thresholds of .50 : .05 : .95.





Evaluation metrics on the test dataset

Method*	Precision	Recall	PMC	PMD	PRM	MAE	MSE
End to End Network	**92.7%**	81%	2.4%	17.1%	97.2%	**17.5**	**18.5**
Multi-Network (Classification Head)	90.4%	**85.2%**	2.3%	24.1%	95%	25.3	27.9
Multi-Network (Regression Head)	90.4%	**85.2%**	2.3%	21.9%	**97.4%**	22.4	23.6

(*) Comparing the complete methods, we observe a significant increase in recall with only a small increase in the distance estimation error.

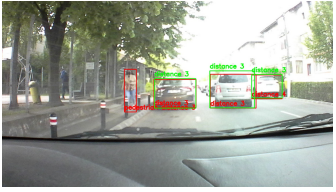
- **Percentage of Missed Category (PMC)**: $PMC = \frac{MC}{TP}$
- **Percentage of Missed Distances (PMD)**: $PMD = \frac{MD}{TP}$
- **Percentage of Relative Mistakes (PRM)**: $PRM = \frac{RM}{TP}$
- **MAE Distance Loss**:

$$MAE_{\text{distance}} = \frac{1}{TP} \sum_{\substack{t=0 \\ \text{lou}\left(\hat{b}_t, b_t\right) > 0.5}}^{\#\text{samples}} \left| \hat{d}_t - d_t \right|$$

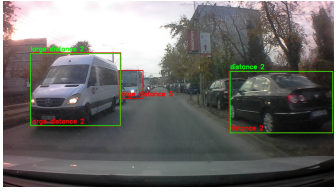
- **MSE Distance Loss**:

$$MSE_{\text{distance}} = \frac{1}{TP} \sum_{\substack{t=0 \\ \text{lou}\left(\hat{b}_t, b_t\right) > 0.5}}^{\#\text{samples}} \left(\hat{d}_t - d_t \right)^2$$

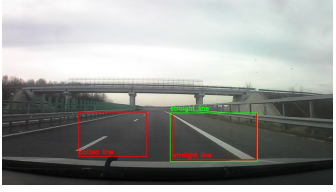
Visual examples of the results



(a)



(b)



(c)

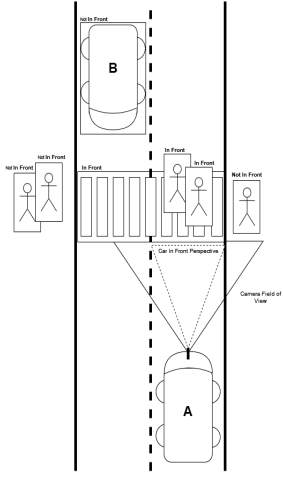
(d)

Red: Multi-Network architecture, Green: End-to-End architecture. As we can observe from the presented images, the Multi-Network approach improves significantly the detection capacity of the system.

Camera horizontal calibration

The next problem we need to solve is detecting the objects that are in front of the vehicle. The objects that are in front of the camera are not necessarily ***in front of the vehicle****.*

In an ideal context, the camera would be always placed/integrated on the middle of the windshield, and after that based on some geometric conditions, we could establish which objects are in front and which of them are not in front.

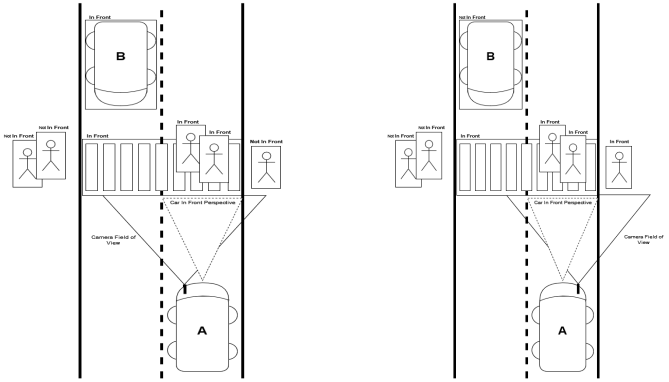


(a) Camera field of view, center positioning

Camera horizontal calibration

In different circumstances, the camera field of view and the in-front perspective of the vehicle might differ.

Those two perspectives are highly influenced by the positioning of the camera on the windshield.



(a) Camera field of view, left positioning. *(b) Camera field of view, right positioning.*

Camera horizontal calibration

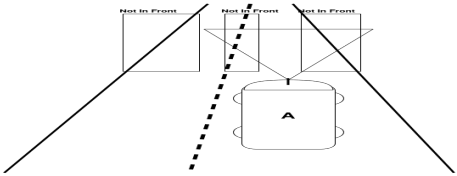
Stacked neural networks

From the previous experiences with stacked networks, we decide to approach this problem based on Machine Learning. For this problem, we use as possible input features the outputs of the previous networks. This way we keep the complexity of the model as small as possible, not affecting the inference time

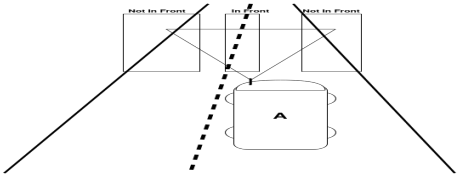
Solution proposed

Considering the positioning of the camera on the windshield as a varying factor that can affect the detection of the in-front objects, we decide to use as an input feature, the camera deviation from the middle of the windshield. This will be further on referenced as the ***calibration offset****.*

Camera horizontal calibration for Lane Departure Warning



(a)



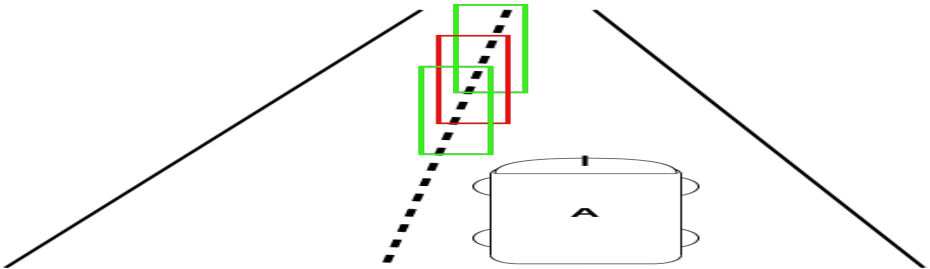
(b)

Example Description

(a) LDW correct non-trigger case.
(b) False positive trigger example from the LDW system.

Lanes interpolation based on features masks

As a fail-safe mechanism, we design an interpolation algorithm for detections based on the previous bounding box coordinates in case of the convolutional neural network fails to detect the object.
We introduce an algorithm based on two steps:
***Accumulation Step** and **Inference Step**.*



Accumulation Step & Inference Step

Algorithm 1: Accumulation Step

```
def AccumulationStep(current_detection, threshold):
    if FeatureMask.active == False:
        /* intiatize the Feature Mask                          */
        FeatureMask = Transform(current_detection)
        FeatureMask.active = True
    else:
        /* add the current detection to the Feature Mask based on IOU  */
        if IOU(FeatureMask.box, current_detection.box) > threshold:
            FeatureMask = Combine(FeatureMask, Transform(current_detection))
            /* update last coordinates                         */
            FeatureMask.box = current_detection.box
        else:
            /* deactivate the mask                             */
            FeatureMask.empty()
            FeatureMask.active = False
    return FeatureMask
```

Algorithm 2: Inference Step

```
def InferenceStep(FeatureMask, current_frame):
    if FeatureMask.active == False:
        return False
    assumed_detection = current_frame [FeatureMask.box]
    S = similarityFunction(FeatureMask, Transform(assumed_detection))
    if similarityConndition (S) == True:
        /* assumed detection is considered as a regular detection  */
        return True
    else:
        /* deactivate the mask                                 */
        FeatureMask.empty()
        FeatureMask.active = False
        return False
```

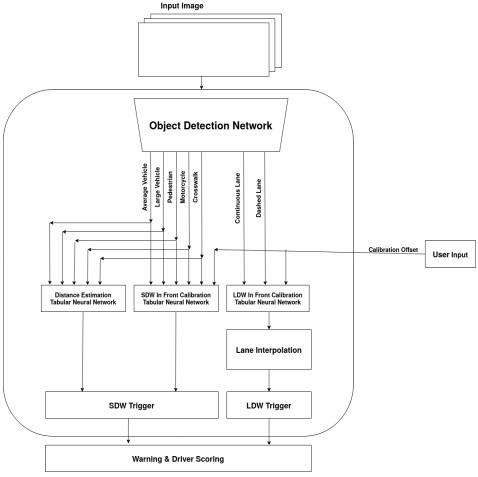
Accumulation Step

To summarize the accumulation step, we initialize the mask based on the highest detection in the current frame and based on the defined Transform and Combine functions, if the IOU condition it's satisfied we accumulate the mask and update the coordinates.

Inference Step

At the inference stage, if the mask is active and we have accumulated a minimum number of consecutive frames, we select the region of interest from the current frame and we check the similarity with the feature mask.

Solution Overview

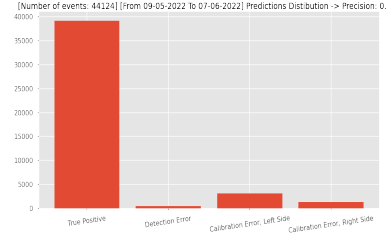


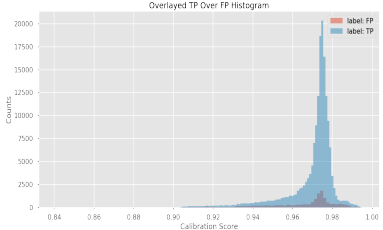
Thesis overview

Results that are presented in the thesis and not described here:

1. Real-life example for the interpolation algorithm with a particular setup
2. An advanced active learning system used for data collection and annotation
3. Experiments with quantized object detection networks for embedded devices (Single Shot Detectors based on MobileNets)
4. An multi-stage pipeline for training the stacked neural networks
5. Evaluating tabular augmentation techniques based on adversarial validation
6. Future possible directions for our solution

Results scaled on multiple vehicles





Described reports

We daily annotate true and false video segments from our clients.

- ***True Positives:*** *detected one or more objects in front of our vehicle, when we are above a certain speed threshold selected based on the estimated distance*
- ***Detection errors:*** *when the object detection network predicts false objects (e.g.: clouds)*
- ***Calibration errors, on the left side:*** *the calibration network detects an object as being in front of the vehicle, but is on the left side of the vehicle*
- ***Calibration errors, on the right side:*** *the calibration network detects an object as being in front of the vehicle, but is on the right side of the vehicle*

Examples from real environment

Raw images and predicted results. *(Left) Raw input image. (Right) Predicted objects in image.*

Examples from real environment

Raw images and predicted results. *(Left) Raw input image. (Right) Predicted objects in image.*

Thank you for your attention