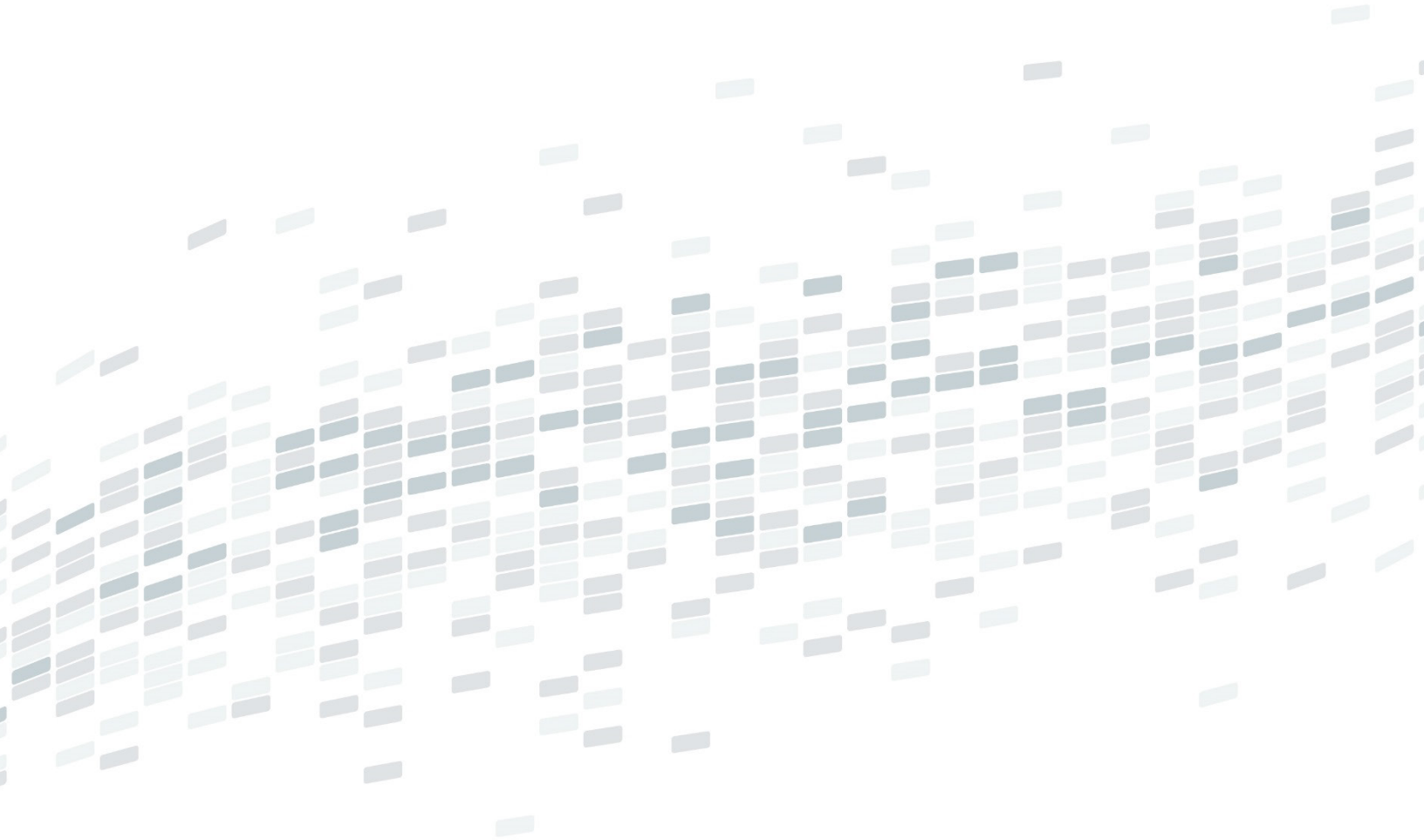
WHITE PAPER
www.visteon.com

Visteon®

Camera-based driver monitoring system using deep learning



Camera based driver monitoring system using deep learning

Nirmal Kumar Sancheti
AllGo Embedded Systems
Bengaluru, India
nirmal@allgosystems.com

Krupa H Gopal
AllGo Embedded Systems
Bengaluru, India
krupahg@allgosystems.com

Manjari Srikant
AllGo Embedded Systems
Bengaluru, India
manjari@allgosystems.com

***Abstract* — Driver Monitoring is emerging as an essential requirement for Advanced Driving Assistance and Autonomous Driving systems. In this paper we propose a real-time, IR camera-based driver monitoring system. Basic driver monitoring features include head tracking, gaze tracking, eye state analysis – blink rate, blink duration, eye open/close all of which can be used to implement driver safety applications like driver distraction and driver drowsiness detection. We propose a system where all these modules have been developed using deep learning which has made the solution more robust to different ethnicities, gender, lighting conditions and occlusions. We have also optimized the solution to run on any embedded platform (ARM, DSP, ASICs etc) without the need of GPU or cloud support during runtime. This helps in lowering power consumption and cost making the solution amenable for use in automotive. An implementation of this system is available as part of the *See 'n Sense* DMS solution from AllGo.**

Keywords—Deep learning, CNN, Driver Monitoring System, Face detection, Gaze estimation, Head pose estimation, Eye state analysis

I. INTRODUCTION

Camera based driver monitoring systems (DMS) can help in detecting driver drowsiness and distraction, thus playing a crucial role in ensuring driver safety. DMS solutions have traditionally been developed using Computer Vision/Image Processing approach. This approach needs fine tuning of various parameters based on different conditions like lighting, ethnicity etc. It is extremely difficult to tune such solutions generically to fit all conditions. More recently, conventional machine learning approaches have been used, where the researcher must decide what features in the data are important and then let the machine learn based on those features. This requires a lot of expertise in that area as well as a lot of research in picking these features. As compared to these approaches, the Deep Learning based approach takes a holistic view of the data and derives features from the data on its own, removing the dependency for hand-crafted features by humans. Deep learning is a subset of machine learning and consists of neural networks which have been inspired by the functioning of human neural system. These networks are very versatile and generalize very well in different conditions.

Convolution Neural Network (CNN) is a special type of deep neural network usually applied to images. The functioning of the network is based on the way our visual cortex functions. Research shows that mammal brain cells consists of two different types of cells. The simple cells which can detect shapes within a restricted area of interest and complex cells which have larger receptive field.

CNNs emulate this by using filters and depth of the network. Since ours is a camera-based DMS solution, it involves processing of the images received from camera and CNNs are used for this purpose.

The basic building blocks for driver safety applications are face detection, head pose estimation, gaze estimation, eye status detection – blink rate, blink duration, open/close status. Deep Learning, being computationally intensive, will typically require a GPU / high end CPU to run real time. In this paper, we present the implementation details of the basic blocks of a

Driver Monitoring System using Deep Learning that runs real time on an embedded platform without the use of GPU / high end CPU.

The organization of the paper is as follows. Section II highlights the approach we have taken to implement the basic blocks of DMS and Section III gives the details of data collection. Section IV talks about how to optimize DMS on an embedded platform and section V describes our commercial offering *See 'n Sense*.

II. Our Approach

"Fig. 1", describes the basic building blocks of our DMS, the outputs of which can be used by an application to implement driver safety solutions. We have implemented these building blocks of DMS using CNNs which have provided better results compared to conventional image processing/computer vision-based implementation.

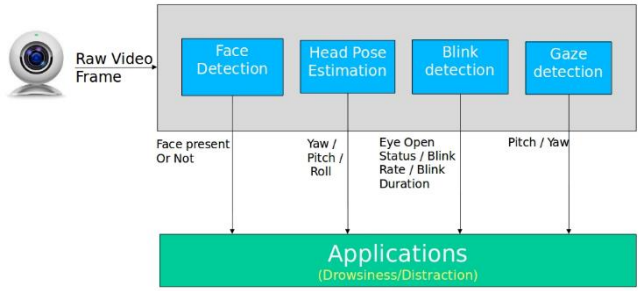


Fig. 1. Block diagram of DMS

A. Face Detection

The first problem in any driver monitoring solution is to detect the face of the driver. Object detection and classification is a much-researched field in deep learning thanks to the incredible success of Alex Net in 2012 on the *ImageNet Challenge*. Object detection and classification consists of two problems – one is localizing the object/s and the second is classifying the object/s. We have treated face detection as an object detection and classification problem. By training a deep neural network with face annotated images, we were able to build a face detector which is robust against various lighting changes, ethnicities, occlusions and expressions.

B. Head pose estimation

Head pose estimation gives the 3D angular orientation of the head in the camera coordinate system. It is represented by three angles – yaw, pitch and roll. Head pose estimation is a challenging task because of large head pose variations and other environmental factors such as lighting, occlusions and expressions. We have developed a CNN based head tracking system using a single camera which takes the detected face region of the image as an input and gives out 3D angles of the head as an output.

C. Gaze estimation

Gaze estimation becomes very important when it comes to detecting distracted driver. Our deep learning-based network is a person-independent, non-intrusive gaze estimator. The gaze estimation network has been trained to work on each of the eye regions independently. The network also takes head pose angles as an input to compute the gaze vectors for each of the eyes. These two vectors can later be combined using simple geometry to get the point of intersection of the two gaze vectors.

D. Eye state analysis

Eye state analysis gives out information related to blink (rate and duration) and eye status (open or close). These parameters become very important in detecting a drowsy driver. By running a non-intrusive calibration procedure for a few seconds when the driver enters the car, blink rate and duration are calibrated for each person. Appropriate deviation from this is then used to signal warnings.

Blink detection as well as eye open/close detection has been implemented using a relatively smaller neural network. Eye open/close is a binary classification problem. Blink detection is a little bit more complex and involves the analysis of the past few frames to detect a blink.

III. Data Collection and Training

For our solution, training happens offline using GPUs and CPUs. The training can take anywhere between a couple of hours to a couple of days depending on the amount of data with which it is being trained and the memory and speed of the machine on which it is being run. It is an onerous task to obtain the data and the ground truth required for training. The quality of the solution is heavily dependent on the quality of the training with which the model is generated.

We have collected data from people of different ethnicities, under different lighting conditions, with different expressions, with various occlusions like sunglasses, scarves, glasses, caps etc. Each module will require a specialized way of collecting and annotating data.

For face detection, data collection will involve capturing faces under all poses, distances, illumination and the data annotation is done manually by annotating the region where the face is present with a bounding box.

For head pose estimation, there is no easy way to collect data and annotate the images offline. Hence, while collecting data, head pose annotation must be done by auxiliary means. There are multiple ways of annotating head pose while collecting data – manually or by using external sensors which can be magnetic based or optical based. Manual annotation is prone to a lot of errors. Magnetic based approach has a limitation that it is sensitive to the presence of metals. We have picked an optical sensor-based ground-truthing mechanism for head pose estimation.

Gaze estimation annotation involves both manual as well as automated methods. Since gaze estimation ground truth requires head pose annotation as well, the procedure used to

annotate head pose will be used together with annotating the gaze direction of the person. A bunch of points are displayed on a screen and the person whose data is being collected is asked to look at each point with various head poses. The person's 3D position of the eye as well as the 3D position of the point being looked at will be measured to get the gaze vector for annotation.

Blink detection and eye state detection annotations are done offline by manually annotating the images. Collection of data should involve all possible scenarios like slow blinks, fast blinks, different blink rates, partial eye closures – at different head poses and lighting conditions.

Once the data and ground truths are ready, to increase the amount of data and variation amongst them, a lot of data augmentation is done before training (some of them being – random cropping, flipping images etc).

IV. Deep Learning on Embedded Platform

Any deep learning solution is computationally intensive which calls for using GPUs or high-end CPUs. We optimize our models heavily to make it run on low cost embedded platform. Below are some of the measures taken to optimize performance for embedded targets.

A. Picking the right network/model

The deep learning model should be picked based on the platform that will be used for inference, the data available for training and the final accuracy desired. If the inference is run in a PC environment with high end CPUs/GPUs, the complexity of the model will not be a concern. Whereas, in an embedded platform, the complexity of the model gains importance. The model which has the least complexity and yet gives desired levels of accuracy should be picked.

For example, in an object detection problem, Inception network or Resnet-50 network will work very well but are computationally intensive and won't be feasible to run on an embedded platform at real-time. A network like SSD or YOLO is a much better choice in terms of complexity. The accuracy might be lower than that of the bigger networks, but compared to traditional computer vision techniques, these win by a large margin.

We have selected the network models carefully and further optimized using pruning and quantization as discussed below.

B. Optimizing the model

Once a model is picked, it needs to be trained with the training data. The trained model can then be optimized by post training techniques like quantization of weights. The quantized model will likely give a lower performance than the full precision version. Retraining the network with the quantized weights derived from the full precision version as the starting point will improve the accuracy.

Another optimization can be careful pruning of the network to reduce the number of weights and reduce the complexity of the network. Another way to lower the complexity of the model is to use newer and better techniques of deep learning like replacing convolution layer with a depth-wise convolution layer.

When running the inference of a model on an embedded platform, the bottleneck is usually the loading of weights in the RAM. Grouping of data which utilize the same weights of the model and processing them batch-wise will result in fewer cache misses.

C. Optimization for embedded platform

Picking the right deep learning tool is very critical for development as well as deployment. ARM offers ARM Compute library (ACL) which is a collection of low-level software functions optimized for ARM Cortex CPU and ARM Mali GPU architectures, targeted at a variety of use-cases including: image processing, computer vision and machine learning.

For our solution, we chose TensorFlow because of its advantages when it comes to training and deployment. TensorFlow has support for distributed computing and good graph visualization tools which becomes critical especially during training. It has faster compilation time than most of the other libraries and provides both C++ and python APIs. It also comes with an optimized, light weight inference library (Tflite) which is best suited for embedded platforms.

In order to make most of the parallelism offered by multi core processors, we can run inference of different modules in different threads. This needs to be done tactically as some of the modules are dependent on the previous modules' output.

V. See 'N Sense

Our current driver safety modules described here are available commercially as part of our *See 'n Sense* offerings. *See 'n Sense* also includes Cabin monitoring modules which are under development. These modules include Face Recognition, Emotion Detection Child detection and Activity detection (Eating, Smoking, Using phone etc.). *See 'n Sense* is currently available on ARM/iMX platform. We have plans to make it available on DSPs and other special architectures.

VI. Conclusion

We have proposed a driver monitoring solution based on deep learning that can be run locally on any embedded platform. In our experience this gives better quality of results compared to conventional approaches.

We expect a huge increase in deployment of Deep Learning based Artificial Intelligence solutions in the coming years and every field would be influenced by it.

References

[1] Ian Goodfellow; Yoshua Bengio; Aaron Courville. Deep Learning

[2] Jonathan Huang; Vivek Rathod; Chen Sun. Speed/accuracy trade-offs for modern convolutional object detectors

[3] Francisco Vicente; Zehua Huang; Xuehan Xiong; Fernando De la Torre; Wende Zhang; Dan Levi. Driver Gaze Tracking and Eyes Off the Road Detection System

[4] Rajeev Rajan;Vishal M. Patel;Rama Chellappa. HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition

[5] Sachin Sudhakar Farfade; Mohammad Saberian; Li-Jia Li. Multi-view Face Detection Using Deep Convolutional Neural Networks

[6] Alberto Fernández; Rubén Usamentiaga; Juan Luis Carús; Rubén Casado. Driver Distraction Using Visual-Based Sensors and Algorithms

[7] Javier Jim´enez-Pinto; Miguel Torres-Torriti. Optical Flow and Driver's Kinematics Analysis for State of Alert Sensing