

Advanced Driver Assistance System based on Intelligent Video Analytics on the Edge

John Stephen

Jeremiah Lewis

Akbar Khader

Andreas Bruenner

MSDS 498, Spring 2023, DL-Sec56

03 June 2023

Table of Contents

1	Executive Summary	3
2	Project Background, Objective, and Business Case.....	3
2.1	Background.....	3
2.2	Objective.....	3
2.3	Business case	4
3	Proposed System.....	4
3.1	System Setup.....	4
3.2	Supported Detection Scenarios	4
4	Technology and Tooling.....	6
4.1	OpenVino and Intel NCS2	6
4.2	NVIDIA Jetson Orin	8
5	Intelligent Video Analytics Pipeline.....	9
5.1	AI Inference Components and Sequence	11
5.2	Flow Diagram and Data Transformations.....	12
5.3	Parallelization of Execution	13
6	Data Profiling.....	14
6.1	Face Detection	14
6.1.1	Overview.....	14
6.1.2	Image Data	15
6.1.3	Labels and Annotations.....	15
6.2	Facial Landmark Detection, Head Pose Estimation, Gaze Estimation	18
6.2.1	Overview.....	18
6.2.2	Image Data	18
6.2.3	Labels and Annotations.....	18
6.3	Open-Closed-Eyes Detection.....	19
6.3.1	Overview.....	19
6.3.2	Image Data	20
6.3.3	Labels and Annotations.....	20
7	Hardware System.....	21
7.1	Components and Assembly.....	21
7.2	Car Installation.....	22
8	Software System	24
8.1	Intelligent Analytics Pipeline	24
8.2	Model Selection	27

8.2.1	Origin	27
8.2.2	Face Detection.....	27
8.2.3	Facial Landmark Detection.....	28
8.2.4	Closed Eyes Detection	28
8.2.5	Head Pose Estimation	28
8.2.6	Gaze Estimation	29
8.3	Model Deployment	29
8.4	System Accuracy.....	31
8.5	Issues.....	31
9	Results.....	32
9.1	Effectiveness of the System	32
9.1.1	Gaze Estimation Results on Live Data.....	32
9.1.2	Distracted Driving.....	32
9.1.3	Driving with Closed Eyes	33
9.2	System Throughput.....	33
9.2.1	Model Throughput	34
9.2.2	Pipeline Throughput.....	35
10	Conclusions and Outlook.....	35
11	References.....	37
	Appendix.....	39
	A1. Verbose output of a custom TensorFlow 2.x model conversion to its OpenVINO IR	39

1 Executive Summary

This document presents the project blueprint for developing and implementation details for an Advanced Driver Assistance System (ADAS) based on Intelligent Video Analytics (IVA) which can be deployed as a self-sufficient, and model-agnostic unit in motor vehicles such as cars, vans, and mini trucks. The project aims to reduce the number and severity of traffic accidents caused by distracted driving, one of the main causes of traffic accidents, and to contribute to the WHO's ambitious goal to cut traffic accidents in half by 2030 as filed via resolution A/RES/74/299. Reversing the worsening trends with respect to traffic fatalities and injuries, however, requires an effective, versatile, affordable, and easy-to-use system. The main audience for the proposed ADAS is insurance companies, which will not only be able to significantly reduce settling costs but also increase their competitiveness by being able to offer cheaper contracts to customers due to a significantly reduced probability and severity of accidents.

2 Project Background, Objective, and Business Case

2.1 Background

According to (World Health Organization, 2021) and (World Health Organization, 2022) approximately 1.3 million and 50 million people (about twice the population of Texas) die or are injured each year, respectively. Most traffic injuries are, hereby, the leading cause of death for children and young adults aged between 5 and 29 years. Furthermore, the leading majority, approximately 93%, of the world's fatalities are happening in low- and middle-income countries, despite those countries only accounting for around 60% of the world's vehicles. A likely reason for this might be the insufficient driver assistance of smaller or cheaper vehicles commonly sold in such countries. Apart from the severe human impact of traffic injuries, there is also a severe economic impact. Road traffic crashes are estimated to cost most countries 3% of their GDP. Those costs are then passed on to insurance companies, which regulate the damage. One of the major risk factors for traffic accidents named by the WHO is distracted driving. The use of mobile phones while driving is a growing concern and driving while using mobile phones has a four times higher probability of being involved in a traffic accident. Other causes for being distracted might be attending to children in the backseat, interacting with a car's onboard system such as the radio or navigation system, smoking, or grabbing utilities. This results in reduced reaction times, the inability to keep the correct lane, and to keep sufficient distance from proceeding cars. Traffic accidents caused by distracted driving are not limited to low- and middle-income countries. According to the traffic safety facts research note by the (Department of Transportation, Washington, DC: NHTSA, 2021) distracted driving alone in 2019 in the United States was the cause of 9% of fatal and 15% of injury crashes of all police-reported motor vehicle accidents, leading to 3142 people killed and close to half a million people injured. According to the research note, the economic damage is equally severe with annual costs of around (about \$400 per person in the US) per year, with around \$9 billion to be paid by causes of accidents directly. In addition, the trend points towards an increase in severe car accidents in general. As stated in the report by the (U.S. Department of Transportation, 2023) close to 43,000 people were killed in motor vehicle traffic crashes in the United States in 2021. Concerningly, this is the highest number of fatalities since 2005 and represents a 10% increase compared to the previous year. Similarly, the number of people injured in 2021 increased significantly by 9.4% totaling around 2.5 million. Thus, being able to counter the trend and reduce the number of car traffic accidents will have a substantial positive societal and economic impact.

2.2 Objective

This project aims to contribute to the WHO's ambitious target of halving the global number of deaths and injuries caused by traffic accidents by 2030, by providing an affordable, vehicle vendor- and model-

agnostic, Advanced Driver Assistance System (ADAS) based on intelligent video analytics for cars, vans, and light trucks. The system's purpose is the real-time detection of prolonged distracted driving and the raising of alarms to force the driver to refocus on the road and the task of driving so that the probability and severity of traffic accidents are reduced.

2.3 Business case

Insurance companies play a key role in rolling out the system to a large customer base and represent the major target audience of the system since it will help to make their contracts more competitive. Less damage caused by their clients that need settling will drive down costs as well as the insurance costs for their customers. The proposed business case is for insurance companies to bundle car, van, and light truck insurance contracts with the mandatory use of the ADAS. We estimate that around 65% of distracted driving accidents can be avoided due to the use of the system reducing human and economic damage and insurance costs accordingly.

3 Proposed System

3.1 System Setup

The system consists of an edge module, a power supply either in the form of a 12V car power plug or if no 12V power outlet is available in the car a Li-ion battery, and a camera, and runs an intelligent video analytics pipeline to detect distracted driving based on gaze estimation. It is mounted in front of the driver to capture the face while driving. The focal area of the video capture is the driver's face with sufficient slack to support different driver heights. The system can be mounted and operated on either side of the vehicle's cockpit to account for left- and right-hand drive and thus can be used in different regions around the world. The system will work on transient data only and thus does not store captured image or video data. Consequently, the system is compliant with most if not all surveillance regulations across the world.

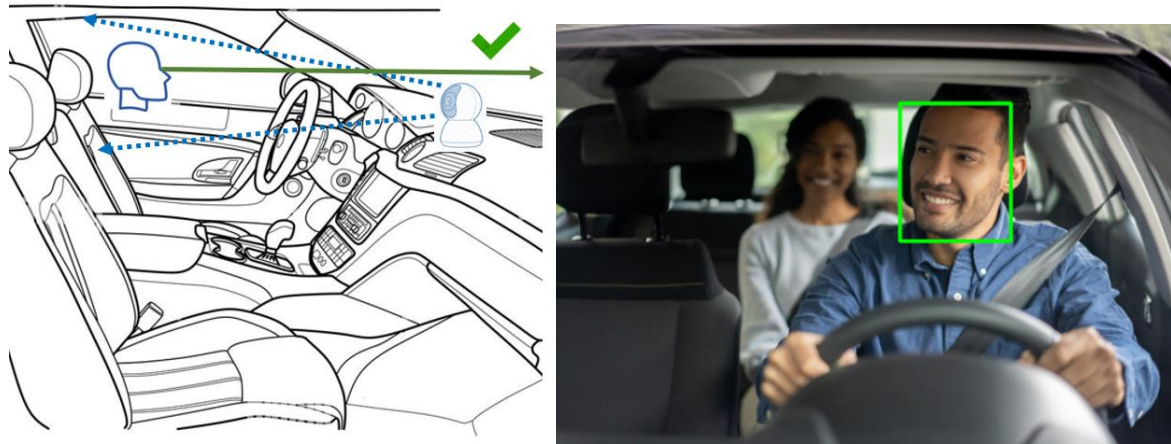


Figure 1. System setup and camera's point of view with the driver's face being the focal area. Right image source: istockphoto.com (modified by adding a bounding box around the driver's face).

3.2 Supported Detection Scenarios

Two scenarios need to be supported by the system to effectively detect the visual distraction of a driver.

1. The driver looks away from the road (down) for a prolonged period.

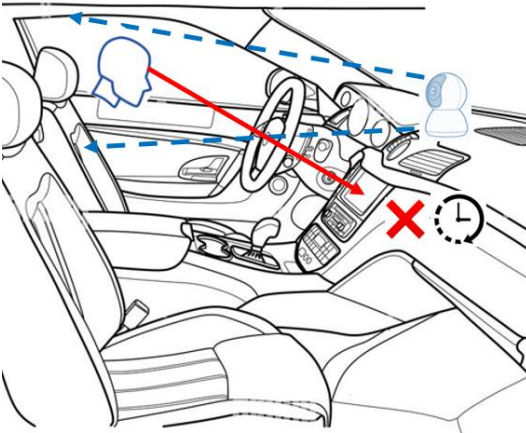


Figure 2. Depiction of the first supported detection scenario. Prolonged gaze away from the road (down).

2. The driver has his eyes closed for a prolonged period, e.g., due to fatigue, microsleep, or being unconscious.

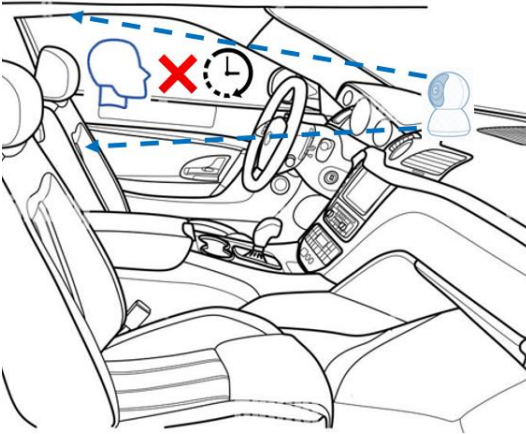


Figure 3. Depiction of the second supported detection scenario. Prolonged closed eyes due to microsleep or being unconscious.

Additionally, the system needs to focus only on the driver, ignoring other passengers if present to increase accuracy and make the system's detection robust.

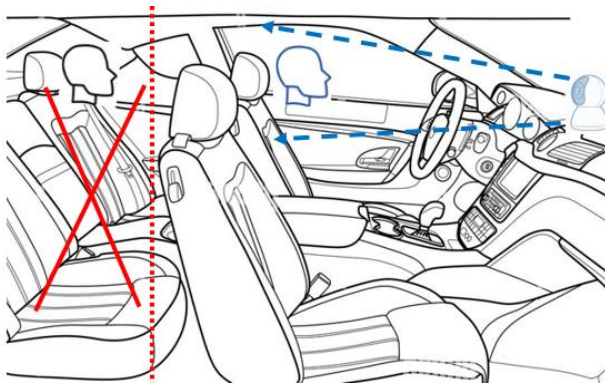


Figure 4. Depiction of the area (backseat) that needs to be ignored by the system to be effective.

For the initial release, the system will notify the driver via an acoustic alarm that is raised from an internal speaker when distracted driving has been detected. For later releases, other actions are conceivable such as decelerating or stopping the car via interfacing with the car's acceleration subsystem to further reduce the probability or severity of an accident. This, however, would require cooperation with car manufacturers.

4 Technology and Tooling

The ADAS as mentioned in the previous chapter must run on an Edge AI hardware/platform so that it can be used in motor vehicles. An a-priori evaluation of edge AI hardware and platforms resulted in two suitable candidates for development and deployment. Please note that the edge AI hardware/platform will mainly be used for video capturing and inference. Model building and training will require sufficiently more computational and energy resources and thus be done on workstations with dedicated GPUs. Early prototypes will be implemented on those platforms followed by a final selection.

4.1 OpenVino and Intel NCS2

The first evaluated platform is OpenVINO, which is an open-source toolkit for optimizing and deploying AI inference on Intel hardware. The main advantage of OpenVINO is the support of most of the deep learning frameworks currently being used by industry and academics such as TensorFlow, PyTorch, ONNX, Caffe, and so forth. OpenVINO provides means to convert custom or pretrained deep learning models from any of the below-listed frameworks to an intermediate format for optimization and deployment on Intel CPUs, integrated and dedicated GPUs, VPUs, FPGAs, and the Intel Neural Compute Stick 2 (NCS2). Moreover, OpenVINO can be used with any of the three most used operating systems, namely macOS, Windows, and Linux. Pretrained models from Intel and the open-source community for various application areas including computer vision are available for direct use, transfer learning, or fine-tuning via OpenVINO's model zoo.

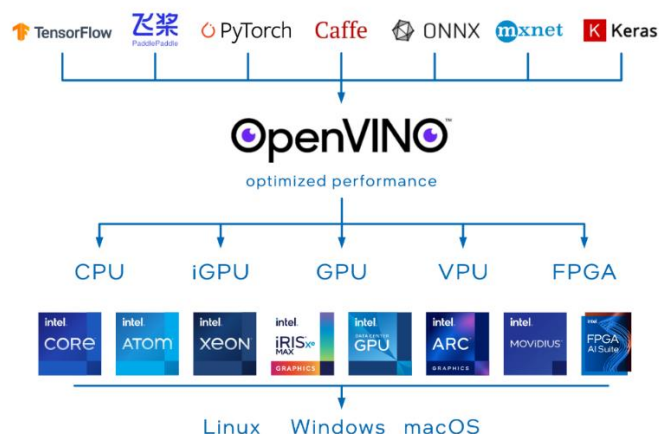


Figure 5. OpenVINO in context. Graphic shows the supported deep learning frameworks, Intel hardware, and supported operating systems. Source: (Intel Corporation, 2023)

The OpenVINO technology stack supports the complete workflow needed for end-to-end edge AI development. This includes the five basic steps planning and setup, selection of a model including training or re-training, conversion of the deep learning model e.g., from TensorFlow 2.x to an intermediate version, specific to OpenVINO (OpenVINO IR), model tuning, and deployment. Especially the deployment step allows for the easy targeting of any of the systems inference device(s). Multiple

models can be deployed on the same or different targets such as CPUs, GPUs, iGPUs, VPUs, FPGAs and Intel's Neural Compute Stick 2 (NCS2).

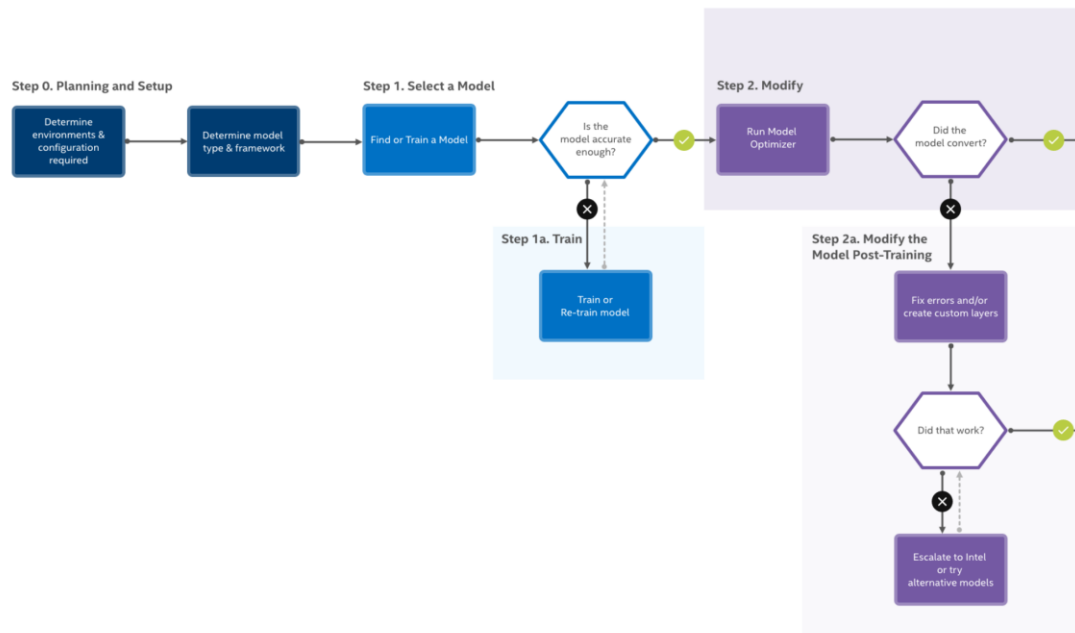


Figure 6. Support of OpenVINO for the edge AI development steps planning and setup, model training and selection, and model conversion. Source: (Intel Corporation, 2021).

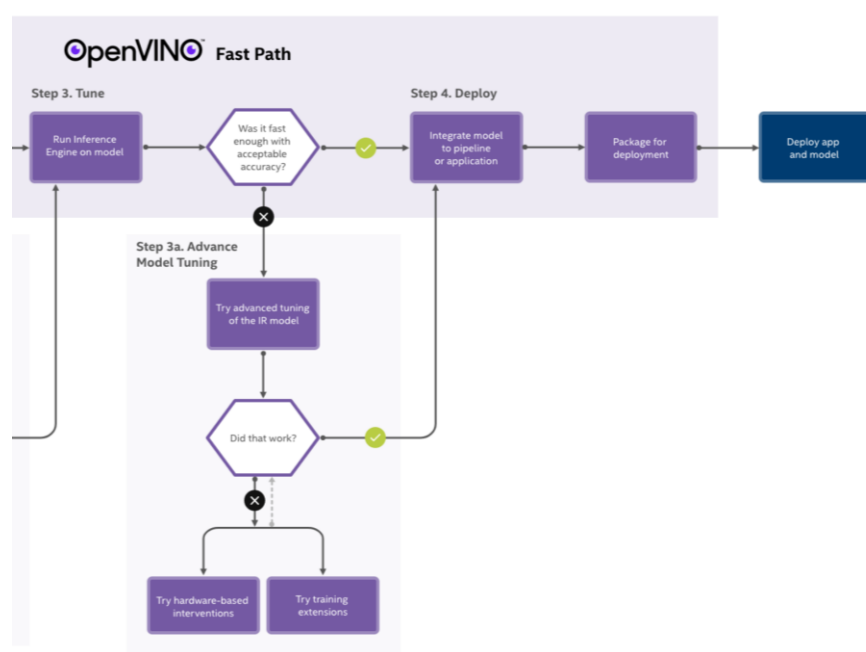


Figure 7. Support of OpenVINO for the edge AI development steps model/performance tuning and model deployment. Source: (Intel Corporation, 2021)

This project uses OpenVINO and the NCS2 for model inference. The NCS2 is a computational unit that requires a host system. Several host system options are supported including single-board computers (e.g., Raspberry Pi), low-power PC setups, or specialized cameras such as the Intel RealSense or Luxonis

DepthAI variants. The various inference tasks can easily be offloaded to either the host's GPU, CPU, or the NCS2. The project's choice for inference is the NCS2 due to affordability, the very small form factor (USB stick), and the minimalistic energy requirements.

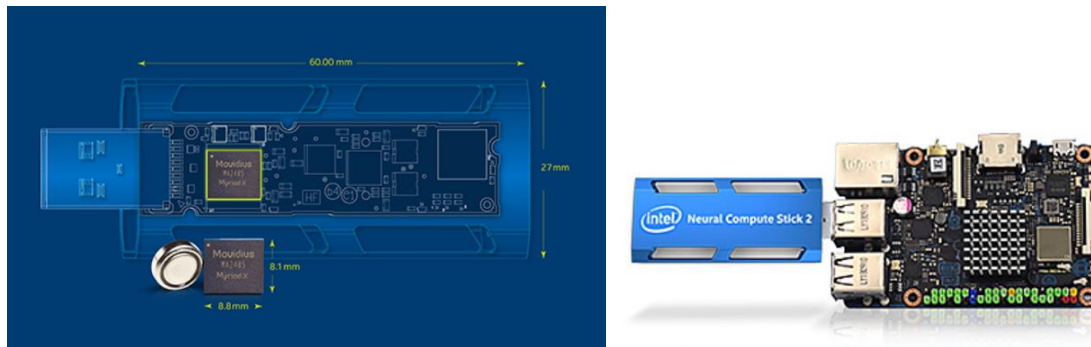


Figure 8. Schematic depiction of the NCS2 on the left and an NCS2 connected to a host system. Source: (Intel Corporation, 2023).

4.2 NVIDIA Jetson Orin

The second evaluated platform is the NVIDIA Jetson Orin platform, which consists of three products supporting small- to large-budget projects. The platform is specially tailored for edge AI tasks and products since it provides dedicated compute units (GPU cores and deep learning accelerators) for deep neural network inference. In addition, the software stack is uniform between the different products making a transition between them straightforward. The software stack, the same as the hardware, is specially optimized for AI inference while at the same time being very energy efficient. The AGX Orin 64GB, which is the topline product for example, achieves up to 275 Tera Operations Per Second (TOPS) with only 60 Watts.

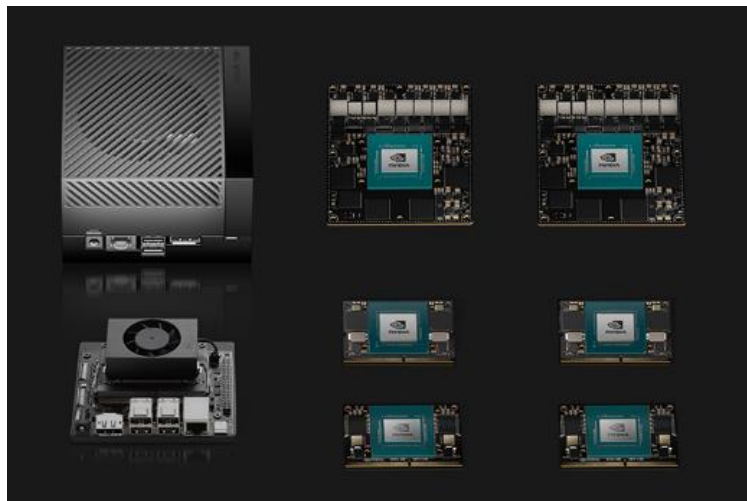


Figure 9. The NVIDIA Jetson Orin product lineup. Source: (NVIDIA Corporation, 2023).

The NVIDIA Jetson platform further provides a software stack that targets, among other things, computer vision projects and as such is a natural fit for the project. Like the OpenVINO toolkit NVIDIA provides a software suite that allows for end-to-end edge AI model development and deployment. The project aims to utilize the NVIDIA Orin NX 16GB for capturing test and demo data.

After the evaluation of the two platforms it was decided to roll-out a system based on OpenVINO and the NCS2 (in conjunction with an affordable host system like the Raspberry Pi 4) since this combination at the time of writing is at least twice as affordable as a system based on Jetson Orin while still being sufficiently fast to support the intended use case in real-time.

5 Intelligent Video Analytics Pipeline

Intelligent video analytics as employed by the ADAS requires a structured flow of data through clearly defined components. This flow is defined as a video pipeline that funnels the input (video stream) from sources like cameras or files through dedicated components. Those components are responsible for pre-processing tasks such as decoding and batching (grabbing a predefined number of images from the stream), AI inference, and post-processing tasks such as tracking, analytics, and composition. The video pipeline's output is actions triggered depending on the pipeline's analytics results and predefined business rules.

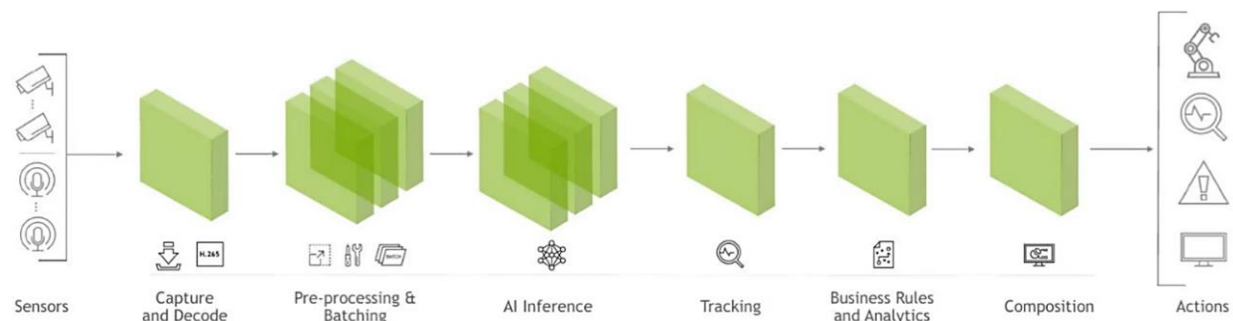


Figure 10. A generic AI video pipeline. Source: (Sinha, 2023)

The video pipeline components needed for the ADAS are explained below.

- **Sensors:**
USB or CSI camera(s) with a supported resolution between 1280x800 and 1920x1080 pixels with at least a frame rate of 30 FPS. The final resolution needs to be determined and is dependent on the performance of the edge platform since larger images take longer to process, possibly conflicting with the real-time requirement of the system.
- **Decoder:**
Decodes the H.264, H.265, or MJPEG encoded video stream to its raw equivalent for further processing.
- **Pre- and Batch-processor:**
Ensures that the color channels are in the correct order, which might either be (A)RGB, RGB, (ABR), or BGR depending on the succeeding component. Grabs single frames from the raw video stream and batches them into suitable chunks e.g., 32 frames per batch. The batch size, among other things, depends on the performance of the used edge platform.
- **AI Inference components:**
Extract valuable information from the stream (batches of frames). For the proposed ADAS, several AI inference components are required. Each AI inference component of the ADAS will be implemented as a deep neural network with required pre- and post-processing tasks.

- **Face detector** to detect the driver's face in a single image. The detected face is defined via the coordinates of a bounding box.
 - **Facial landmark detector** to determine the facial landmarks around the driver's eyes. Those landmarks are then used to determine the coordinates of the eyes as bounding boxes.
 - **Head pose estimator**
Depending on the concrete implementation of the detector succeeding in the pipeline this component is needed to provide head rotation and tilt angles. The angles are potentially used by the gaze estimator component.
 - **Gaze estimator** to determine the gaze of a driver in angles. The angles are used in the business rules and analytics component.
 - **Closed eyes detector** to determine whether the driver's eyes are closed. A flag indicating whether the eyes are closed or open is used in the business rules and analytics component.
- **Tracker:**
Stores information such as face position, gaze angles, and closed eyes indications for a predefined period. For the ADAS this is needed for example to determine when the eyes are closed or whether for example the last 3 seconds the gaze angles were pointing away from the road. In essence, the tracker provides temporal dependencies between single frames.
 - **Business Rules and Analytics Component:**
Contains the predefined period in seconds that the gaze is allowed to wander off the road and that the eyes are allowed to stay closed. Excess of the predefined thresholds will cause a distracted driving indicator to be raised.
 - **Compositor:**
Aggregates the two distracted driving indicators, one from a gaze and one from a closed eyes violation and forward the need to raise an alarm.
Depending on the requirements compositor, analytics component, and tracker might be fused during implementation.
 - **Actor:**
Will raise an acoustic alarm if at least one indicator of distracted driving has been forwarded.

Several options regarding the choice of AI inference components and configurations exist that all result in an effective gaze estimation. For now, however, the configuration presented above is the configuration of choice for the ADAS.

5.1 AI Inference Components and Sequence

The following diagram details the AI inference components and their sequence that will be required for distracted driving detection.

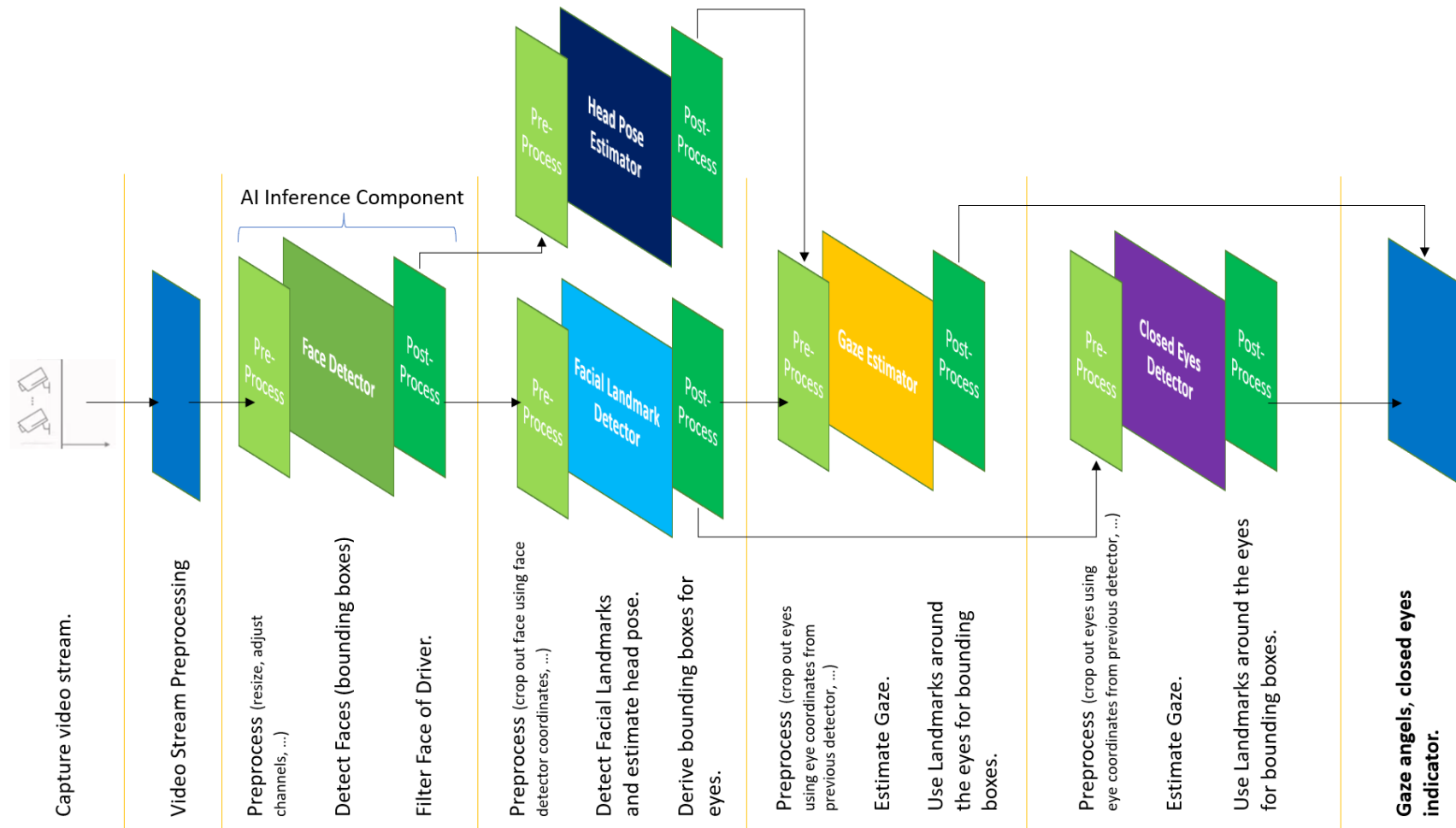


Figure 11. AI inference components with respective pre- and post-processing steps and their intended sequence.

5.2 Flow Diagram and Data Transformations

The following diagram shows the video pipeline components, their functions, required data transformations, and their data exchange.

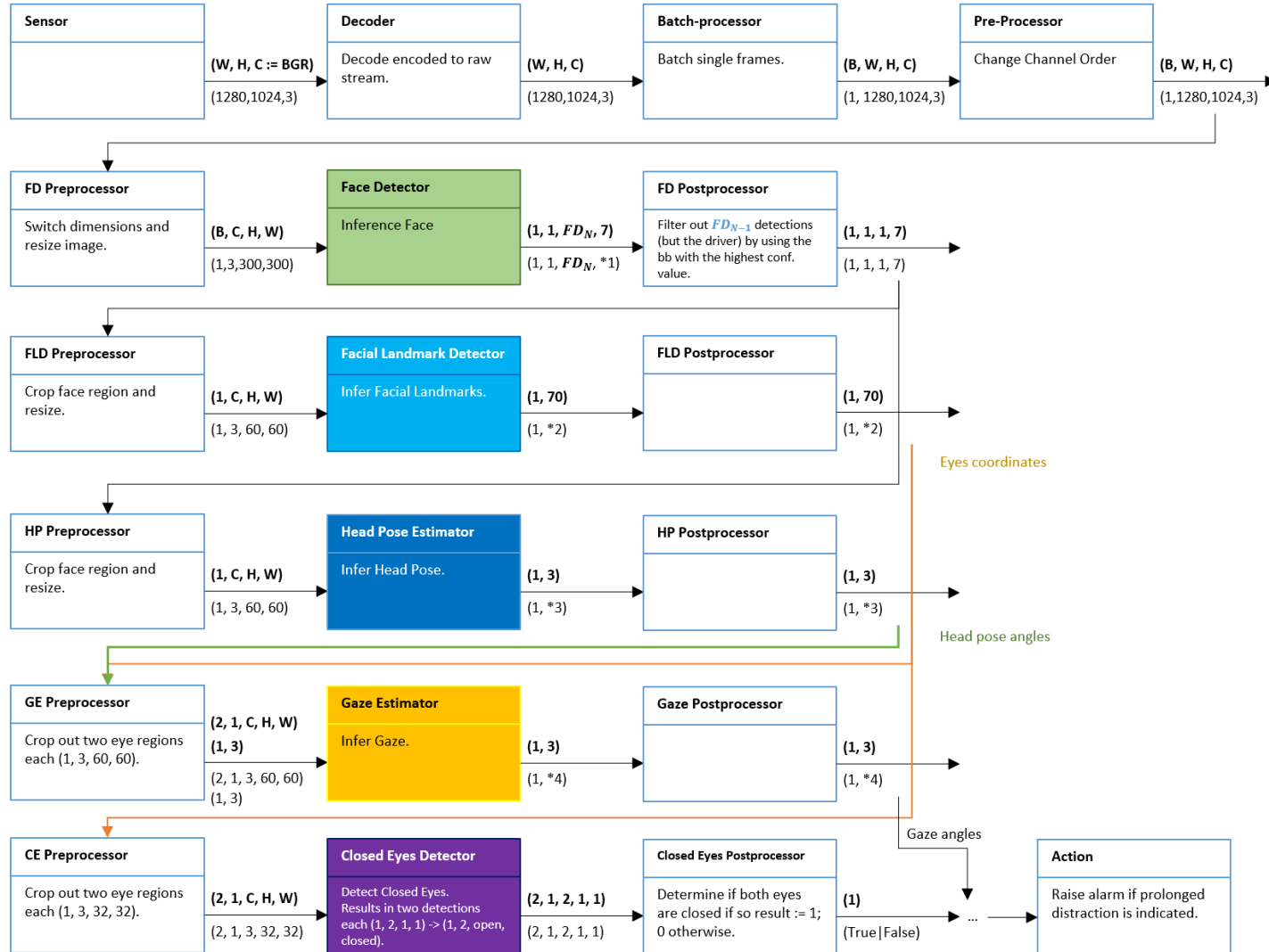


Figure 12. Video pipeline in detail as intended for implementation with OpenVino.

Legend

B: Batch size.
W: Width in pixels.
H: Height in pixels.
C: Channels.
BGR: Blue, Green, and Red Channel.

Foot Notes

(*1): N face detections with each represented as 5-tupel
(image ID, label, confidence, x min, y min, x max, y max).

(*2): 35 facial landmarks each represented as (x, y) coordinates.

(*3): 3 angles. Estimated yaw, estimated pitch, and estimated roll in degrees.

(*4): Non-unit cartesian gaze direction vector pointing to the coordinate (x, y, z).

The concrete dimensions (dimension below each arrow) are assumptions and might vary depending on the concrete architecture and implementation of each neural network. Further, the use of a closed eye detector might be optional, if gaze estimation already covers this use case.

5.3 Parallelization of Execution

The flow diagram of the proceeding section shows that there are some AI components that can be executed in parallel. The main advantage of doing so is a higher throughput reflected by higher FPS. A common drawback of using parallelism is the addition of complexity conceptually and regarding implementation. Consequently, the project will exploit this opportunity only if necessary to counter performance issues. The following diagram shows the two possibilities for parallel execution. The first possibility is the parallel execution of the facial landmark detector and head pose estimator, which both require the face coordinates from the face detector for execution. The second possibility is the parallel execution of the closed-eyes detector and gaze estimator. The gaze estimator requires the head pose angles and the eye coordinates as input, whereas the close eye detector only requires the eye coordinates to execute. The improved approximated execution time in either case when applying parallelization would result in

$$t_{total} = \max(t_{component1}, t_{component2})$$

instead of

$$t_{total} = t_{component1} + t_{component2}$$

when executed sequentially.

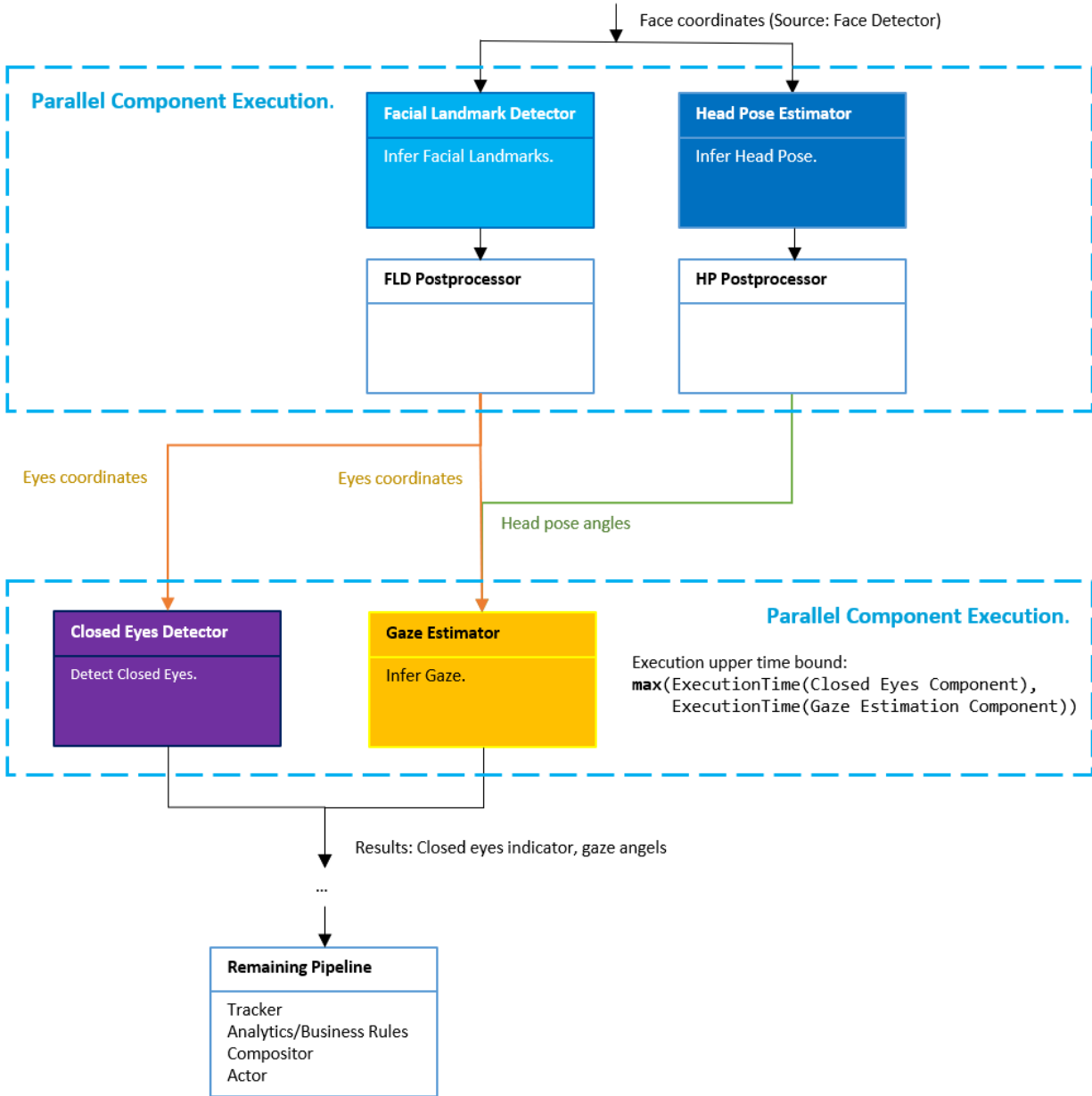


Figure 13. Options for the parallel execution of the video pipeline to improve performance.

6 Data Profiling

Three data sets have been chosen that will be used for training, transfer learning, or fine-tuning required deep learning models if needed. Only three data sets are required for our purposes instead of five since one of the data sets can be used for facial landmark detection, head pose estimation, and gaze estimation.

6.1 Face Detection

6.1.1 Overview

The project uses the WIDER FACE data set by (Yang, Luo, & Tang, 2016) as the primary data set for the training, transfer learning, or fine-tuning of a deep learning model to detect faces. Initially released in 2016, it is still considered one of the largest and most well-curated data sets for face detection. It consists

of 32204 images and 393703 annotated faces. The data set is split into training, test, and validation set with 12880, 16097, and 3226 images respectively.

6.1.2 Image Data

All images have a width of 1024 pixels. The average height in pixels is around 888 pixels with a standard deviation of around 350 pixels. The distribution of image heights is right skewed mainly due to 50 extreme outliers with values of up to 9108 pixels.

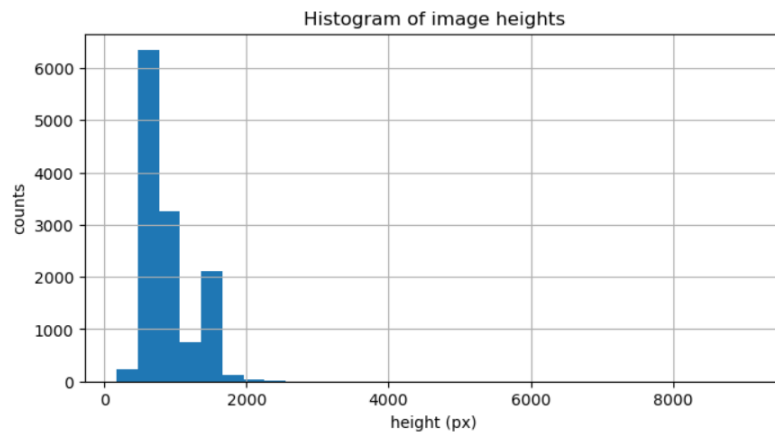


Figure 14. Image height distribution shows a right skewness.



Figure 15. Sample of a height outlier with around 2500 pixels in height.

The 50 outliers will be removed before use.

6.1.3 Labels and Annotations

Each image contains ~12.4 faces on average annotated as bounding boxes. We expect the use of images with multiple faces to add to the robustness of the used model since it will add to the variability concerning occlusion, vantage points, scaling, lighting conditions, etc.







	faces/bbox	faces/blur	faces/expression	faces/illumination	faces/invalid	faces/occlusion	faces/pose	image	image/filename
0		1	False	False	False	0	False		12--Group/12_Group_Group_12_Group_Group_12_108.jpg
		1	False	False	False	0	False		
		1	False	False	False	0	False		
		1	False	False	False	0	False		
		1	False	False	False	0	False		
		1	False	False	False	0	False		
1			11--Meeting/11_Meeting_Meeting_11_Meeting_Meeting_11_700.jpg
		1	False	True	False	2	False		
		1	False	True	False	0	False		
		1	False	False	False	2	False		
		1	False	True	False	0	False		
		0	False	False	False	0	False		
2		1	False	True	False	0	False		28--Sports_Fan/28_Sports_Fan_Sports_Fan_28_290.jpg
		1	False	False	False	0	False		
		1	False	False	False	1	False		
		1	False	False	False	0	False		
		1	False	False	False	0	False		
		2	False	False	False	2	False		

Figure 16. Three samples from the WIDER FACE data set showing the image data as well as the included annotations.

There are in total seven labels/annotation categories.

Name	Description	Data Type
bbbox	The location and size of the bounding box annotating faces.	Float32 four-tuple representing (x1, y1, w, h).
blur	The amount of blur for a given bounding box.	uint8. 0: Clear 1: Normal 2: Heavy
expression	The expression of an annotated face.	bool. 0: Typical 1: Exaggerate
illumination	The illumination of an annotated face.	bool. 0: Normal 1: Extreme
invalid	Indicator whether a bounding box is invalid.	bool. 0: valid 1: invalid
occlusion	The degree of occlusion of an annotated face.	uint8. 0: No occlusion 1: Partial occlusion 2: Heavy occlusion
pose	The pose of the detected head.	bool. 0: Typical 1: Atypical

Around 1.5% of the annotated bounding boxes in the data set are invalid and should be removed.

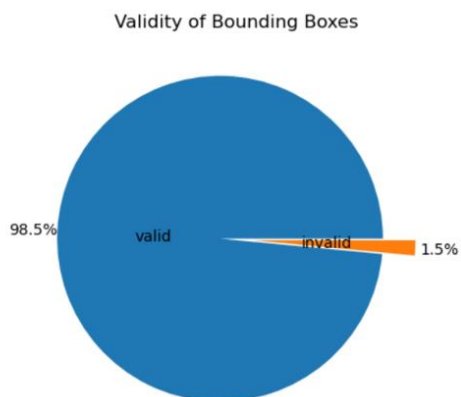


Figure 17. The ratio of valid and invalid bounding boxes shows that around 1.5% of the bounding boxes are invalid.

Images with invalid bounding boxes will be removed before training. The data set exhibits imbalances concerning the amount of blur and occlusion of bounding boxes.

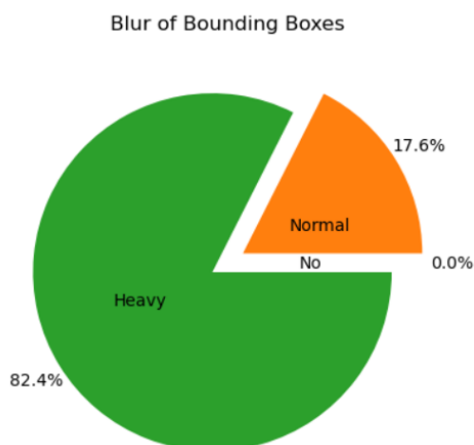


Figure 18. The ratio of bounding boxes with no, partial, and heavy occlusion.

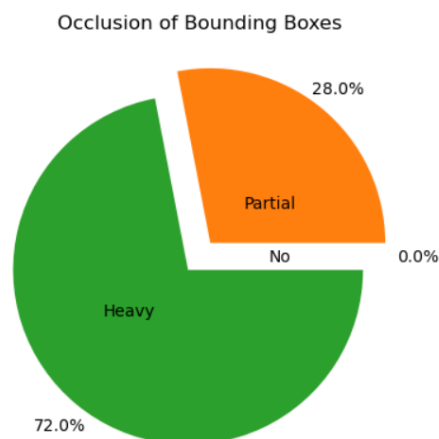


Figure 19. The ratio of bounding boxes with no, normal, and heavy blur.

We will apply operations to deblur heavily blurred images as outlined by (Zang, Kaihao, Ren, Luo, & Lai, 2022) to balance the data set. An alternative to this approach could be data augmentation using translation, zoom, and flipping along the y-axis to increase the percentage of normally blurred images. We do not expect heavy occlusion to negatively impact the training of the face detector since no problem has been reported in that regard in the original article. The only real anomaly of the data set is the annotation of illumination. Not all bounding boxes are tagged correctly regarding illumination and all of those that are tagged are tagged as extremely illuminated. Screening samples manually showed that there is sufficient variability with respect to illumination in the data set. Consequently, illumination labels can be ignored. We plan to remove images with invalid bounding boxes and height outliers as well as resizing the images according to the input layer dimension of our deep learning-based face detector.

6.2 Facial Landmark Detection, Head Pose Estimation, Gaze Estimation

6.2.1 Overview

The project uses the MPIIFaceGaze data set by (Zhang, Sugano, & Bulling, 2019) as the primary data set for the training, transfer learning, or fine-tuning of deep learning models used to detect facial landmarks, estimate head poses and gaze. The data set consists of 37,667 images collected from 15 participants over three months. It provides a large variability in appearance and illumination required to train models robust enough to support different driving times (morning, midday, evening, night), seasons, and so forth. In addition, the distance between the camera and the captured person is like our intended setup.

6.2.2 Image Data

All images are equally sized with a resolution of 1280x720. There are, however, 7 outliers with a resolution of 320x240 pixels, which will be removed before use.

6.2.3 Labels and Annotations

The facial landmark annotations of the eyes will provide the width of the eye regions. Bounding boxes around the eye regions can then be constructed from squares based on the determined width. We will use annotations xy1, xy2, and xy3, xy4 for the left and right eyes respectively. As shown below, the face images are embedded in either black, white, or mixed backgrounds. This needs to be accounted for by cropping out the faces e.g., via min-max thresholding, image segmentation, or alike.

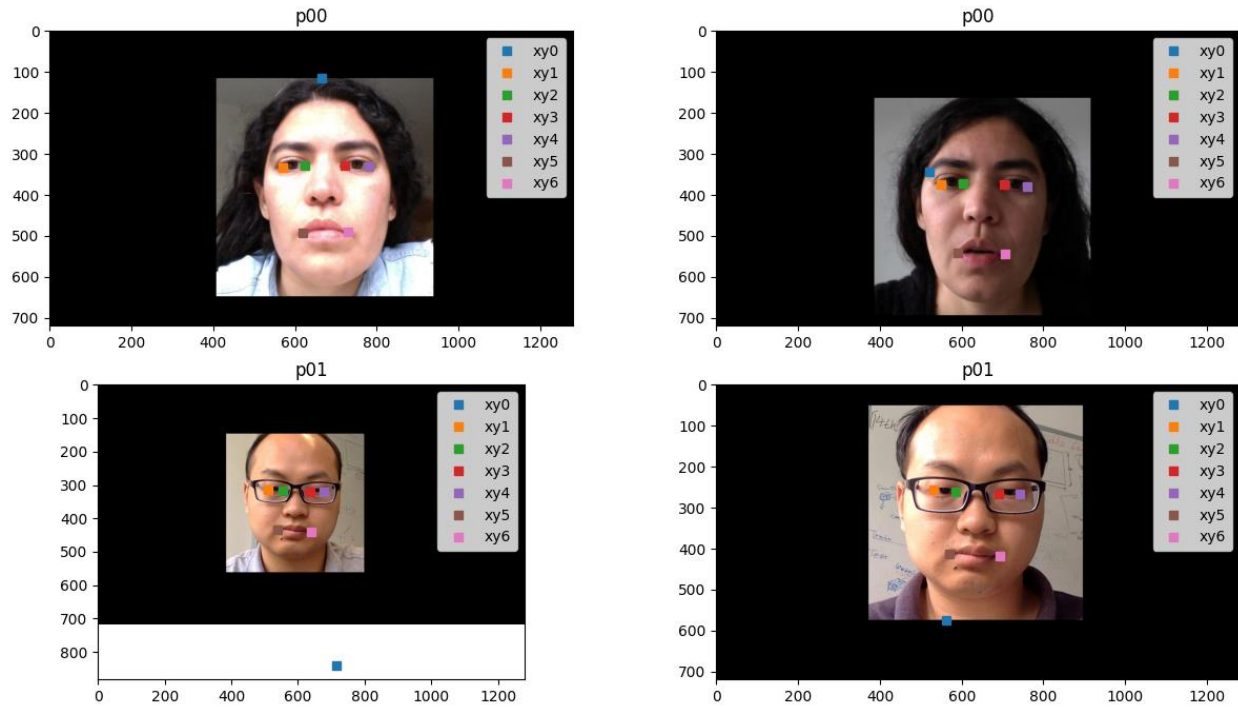


Figure 20. Four samples from the MPIIFaceGaze data set showing six facial landmarks for each of the four faces.

The face position plot below shows a homogeneous distribution of face positions. This is important since we intend to have a centered view of the face with the ADAS as well. There are 9 extreme outliers, which should be removed before use.

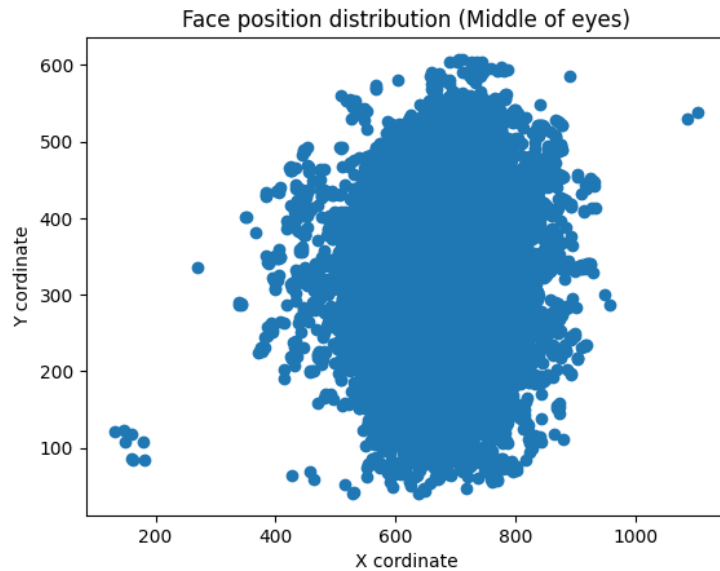


Figure 21. The face position distribution (center between the eyes) shows a homogeneous pattern with 9 extreme outliers.

The gaze location on the calibrated screen shows a homogenous pattern. There are, however, six outliers that exceed the known screen size and that should be removed before use.

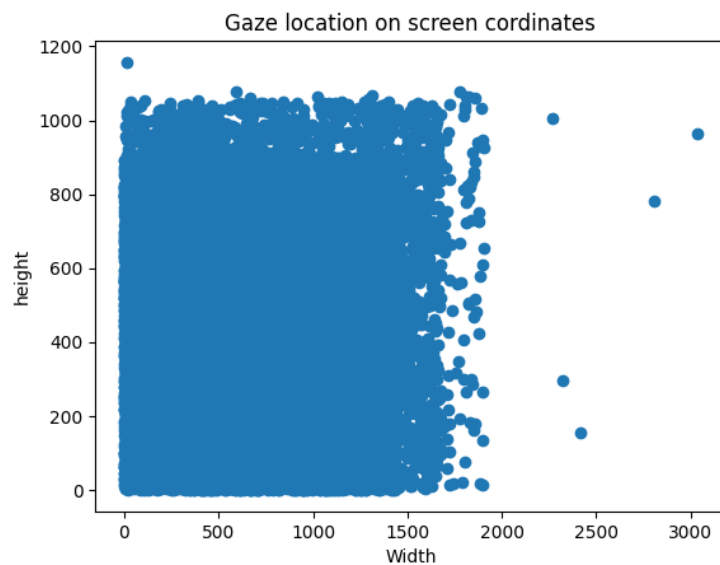


Figure 22. The gaze locations on screen coordinates show a homogeneous pattern with 6 outliers.

6.3 Open-Closed-Eyes Detection

6.3.1 Overview

The project uses the MRL Eye data set as proposed by (Fusek, 2018) as the primary data set for the training, transfer learning, or fine-tuning of the deep learning-based closed-eyes detection model. The dataset encompasses around 85,000 images from a total of 37 individuals of those being 33 men and 4 women. The data set was captured using near-infrared cameras of three types namely, Intel RealSense SR300, IDS imaging, and Aptina Imagin, with 640x480, 1280x1024, and 752x480 pixels resolution respectively.

6.3.2 Image Data

All images are grayscale 217x217 pixel patches, and it needs to be determined if a model trained on grayscale (near-infrared) images provides sufficient accuracy when inferring on colored images.

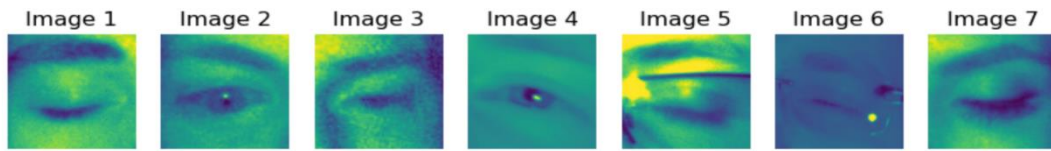


Figure 23. Seven sample images showing the variability of open and closed eyes using the Viridis color map.

There is significant variability in the luminance as illustrated below, which should provide sufficient variability to account for different driving times and seasons.

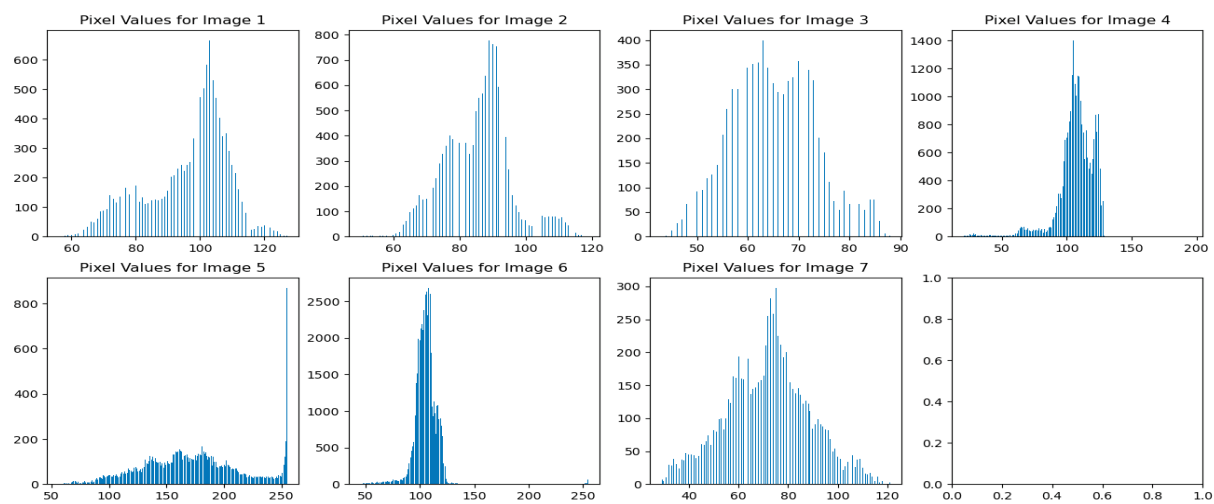


Figure 24. Histograms of seven sample images showing significant variability in luminance.

6.3.3 Labels and Annotations

There are six different labels in the data set apart from the subject ID and image number.

Name	Description	Data
Subject ID	id for identifying person(s)	xxx
Image #	id/# to keep up with number of images taken	xxx
Gender	contains the information about gender for each image (man, woman)	0 – male 1 – female
Glasses	information if the eye image contains glasses	0 – no 1 – yes
Eye State	contains the information about two eye states	0 – close 1 – open
Reflections	three reflection states based on the size of reflections	0 – none 1 – low 2 – high
Lighting Conditions/ Image Quality	image has two states (bad, good) based on the amount of light during capturing the videos	0 – bad 1 – good
Sensor Type	the dataset contains the images captured by three different sensors	01 – RealSense SR 300

		02 – IDS Imaging 03 – Aptina Imagin
--	--	--

The ratio of images showing male and female eyes is imbalanced and should be countered by data augmentation like face detection using translation, zoom, and flipping along the y-axis.

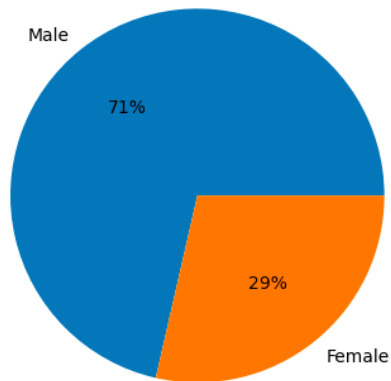


Figure 25. The male to female eye ratio in the data set shows a significant imbalance.

An analysis of the sensor type shows a somewhat balanced data set regarding the Intel RealSense and the IDS Imaging sensor. Aptina Imagin even though listed as a sensor type in the data set was not used for capturing images.

7 Hardware System

7.1 Components and Assembly

The hardware system used to capture videos consisted of

- An NVIDIA Jetson Orin NX 16GB developer board.



Figure 26. The NVIDIA Jetson Orin NX 16GB developer board next to a credit card to illustrate the small form factor.

- An IMX219-77 camera with a diagonal angle of view of 80 degrees and 8 Megapixels.

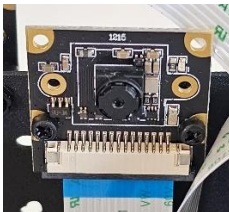


Figure 27. The IMX219-77 camera assembled to a camera holder.

- A 5600mAh Li-ion battery, a battery adapter plate with 12V DC Output, and a DC-to-DC cable. With the chosen configuration it is possible to charge the battery via USB.



Figure 28. Hardware required to allow mobile use e.g., without the need for a fixed power supply.

- A chassis with a camera holder to be able to fix and adjust the camera position and angle.



Figure 29. The assembled hardware of the ADAS running on 12V DC supplied by a Li-ion battery.

The form factor of the NVIDIA Jetson Orin is comparable to a system based on a Raspberry Pi 4 and the NCS2 and was used over the Raspberry Pi 4 due to availability. The first, however, is around twice as expensive. Since the system is intended to be affordable for customers in low- to mid-income countries as well, development of the ADAS focused on the less expensive platform (OpenVINO and NCS2).



Figure 30. The Intel Neural Compute Stick 2 (NCS2) is used for model inference.

7.2 Car Installation

The system was installed facing the driver's face with around 1 meter distance between the camera and the face. The ADAS is not bound to a specific distance from the driver's face and thus supports the installation of the ADAS in all targeted vehicle types from small, mid, and large-sized cars to vans and small trucks with varying degrees of available space.



Figure 31. Installed ADAS. Camera to face distance ~1 meter.

The use of the car's 12V power outlet to power the system is also feasible but was not used since not all vehicle types and models might provide such an option. Offering the system with a 12V power plug instead of a battery, however, should be considered during roll-out.



Figure 32. Placement of the power source (battery).

The chosen camera supports a resolution of up to 4k. Due to the real-time requirement, however, the best balance between inference speed (FPS) and resolution was 1280 by 720 pixels. As illustrated below this resolution offers sufficient image size and quality while at the same time being small enough to execute required analytics sufficiently fast.



Figure 33. Example video capture from the ADAS using a resolution of 1280x720 pixels.

8 Software System

8.1 Intelligent Analytics Pipeline

The preliminary software system initialization consists of the following execution steps:

- Command line argument parsing by the argument parser.
The system reads in essential arguments such as the target device (CPU, GPU, MYRIAD/NCS2), video input (image, video file, video stream), and model definition files (xml files) for each AI component via a launch.json or directly as command line parameters.
- The initialization of the OpenVINO Runtime Core and its use as part of the inference execution models (IEModels). The IEModels wrap and execute the neural networks on input data. Three different IEModels are required:
 - o **Single Input/Single Output (SISO)**
The face detector for example takes a frame as (single) input and returns the position of a detected face (single output).
Since there can never be more than one active driver the face detector employs confidence thresholding, returning only the bounding box with the highest probability of containing a face. The core assumption that the camera is always mainly focused on the driver should hold for any common driving scenario. The facial landmarks detector is also a SISO model.
 - o **Single Input/Multiple Output**
The head pose estimator takes the image data of the detected face as input and returns the estimated head pose expressed as three different angles in degrees (yaw, pitch, and roll).
 - o **Multiple Input/Single Output**
The gaze estimator takes the image data of both eyes and the estimated head pose (angles) as input and returns the estimated gaze angles.
- Call to initialize the video pipeline and start the execution of the pipeline on a single image or video stream. The system was tested with H.264 compressed streams. Uncompressed streams as well as H.265 compressed streams, however, should be supported as well.

The following components are essential to the system and their initialization as part of the ADAS is illustrated in the following sequence diagram:

- Argument parser, to read configuration settings and other arguments.
- Image capture, which grabs single frames from a video stream for further processing.
- AI components, representing the “intelligent” part of the intelligent video analytics pipeline.
 - o Face detection component
 - o Facial landmarks detection component
 - o Closed eyes detection component
 - o Head pose estimator component

- Gaze estimation component
- Result renderer visualizes results as annotations directly in frames.
This component can be deactivated after the final system test and before system roll-out and should further improve system throughput (higher FPS).
- Pipeline wrapper and its contained steps representing the actual video pipeline.

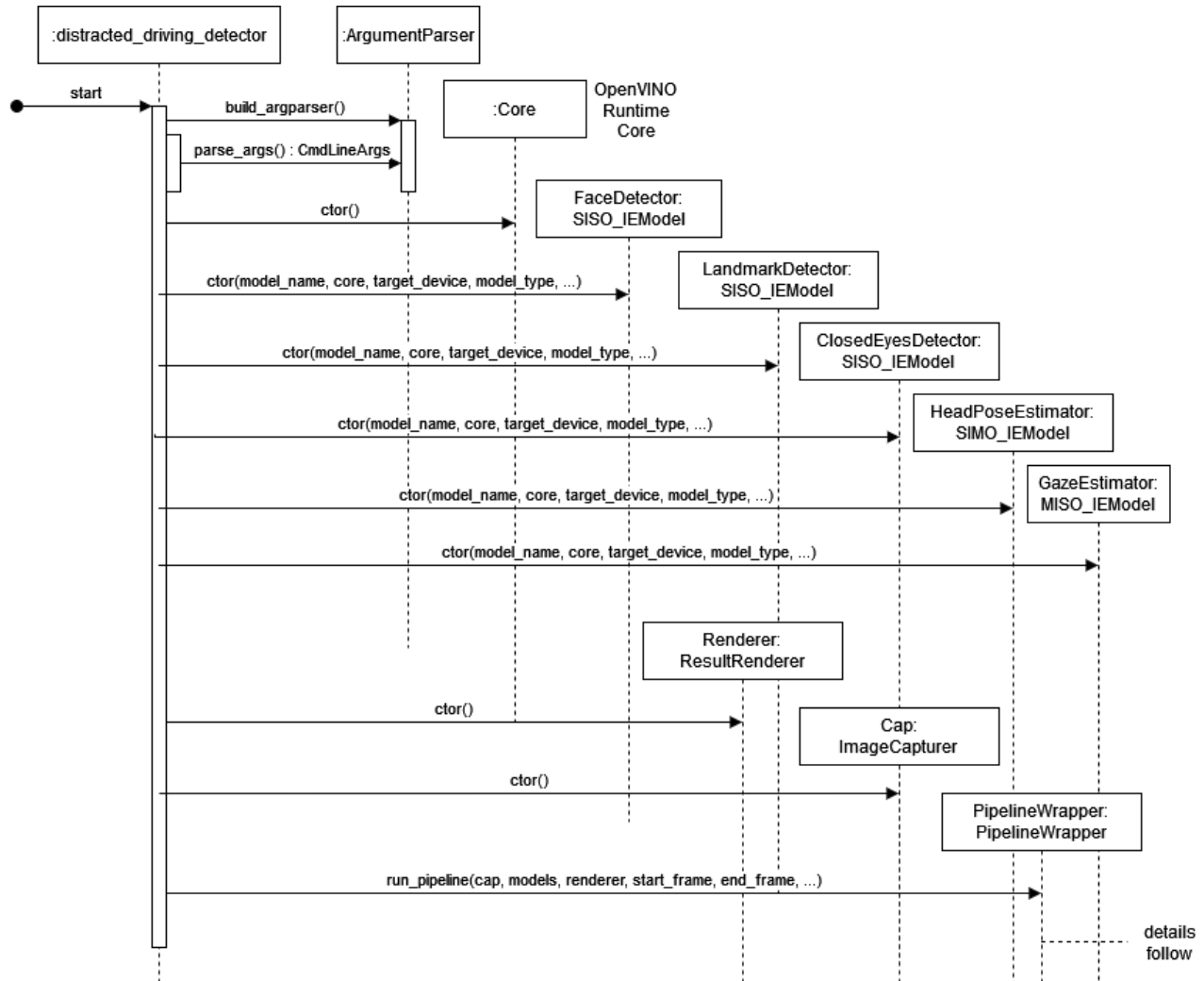


Figure 34. Preliminary system initialization.

The following sequence diagram shows the execution sequence of the video pipeline initialization and execution. After initialization of the asynchronous pipeline, the single steps are first initialized and then added to the pipeline. This includes the steps encapsulating the AI components, the tracker, and the result renderer. Afterwards the pipeline is executed until no more data is to be processed. As soon as no more data needs to be processed by the pipeline, it is closed and statistics such as the execution times of the components in milliseconds and performance in FPS are gathered and reported. Those statistics were crucial during development to evaluate if the real-time requirement of the system is still met.

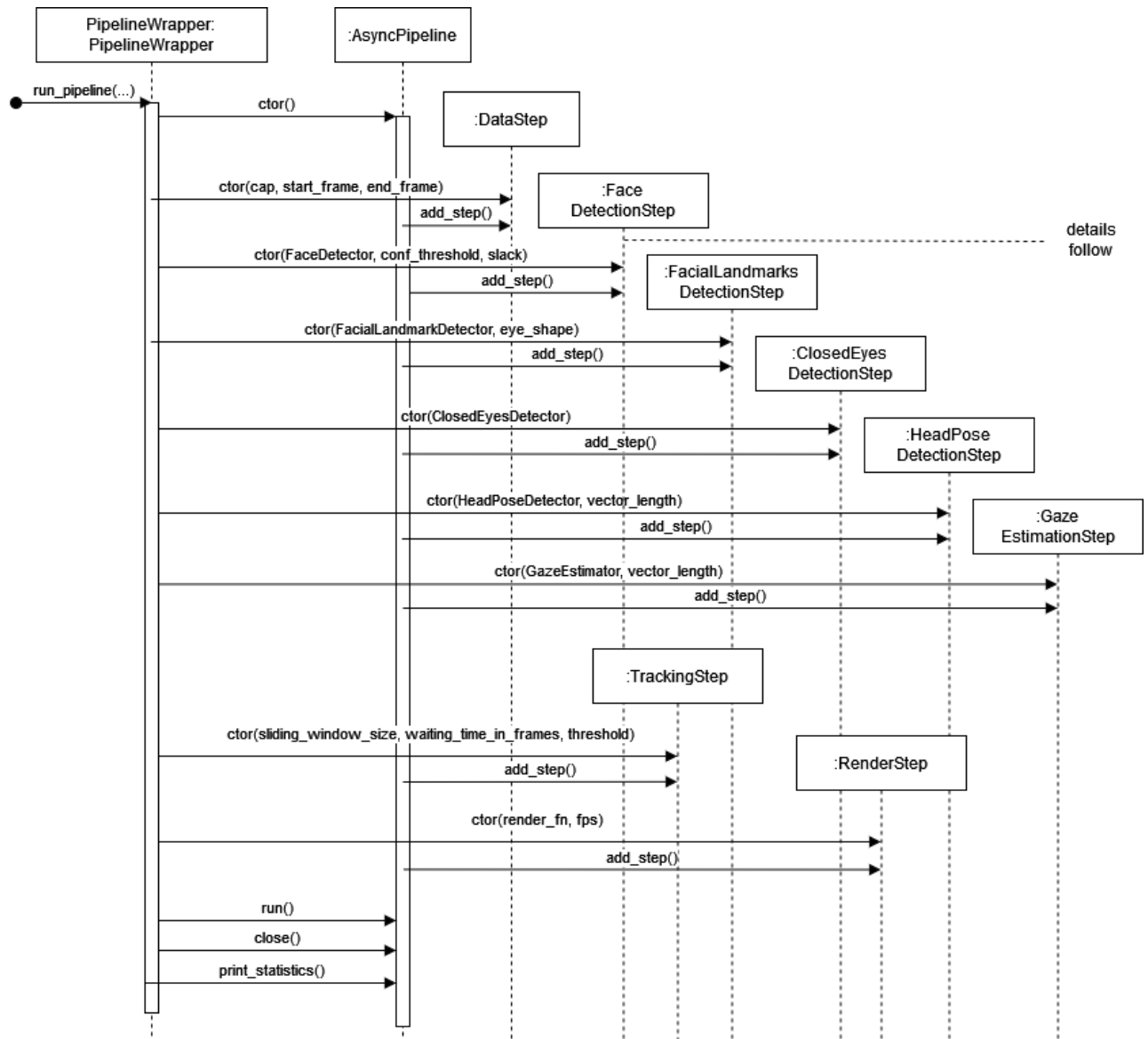


Figure 35. Video pipeline initialization and execution steps.

Each pipeline step contains a detector and an asynchronous wrapper. The detector is used for input pre- and output post-processing. The asynchronous wrapper contains the model (neural net, wrapped as an IEModel) for inference on a given input. Each step produces a result, which is added to an aggregate result object to share intermediate results between the steps.

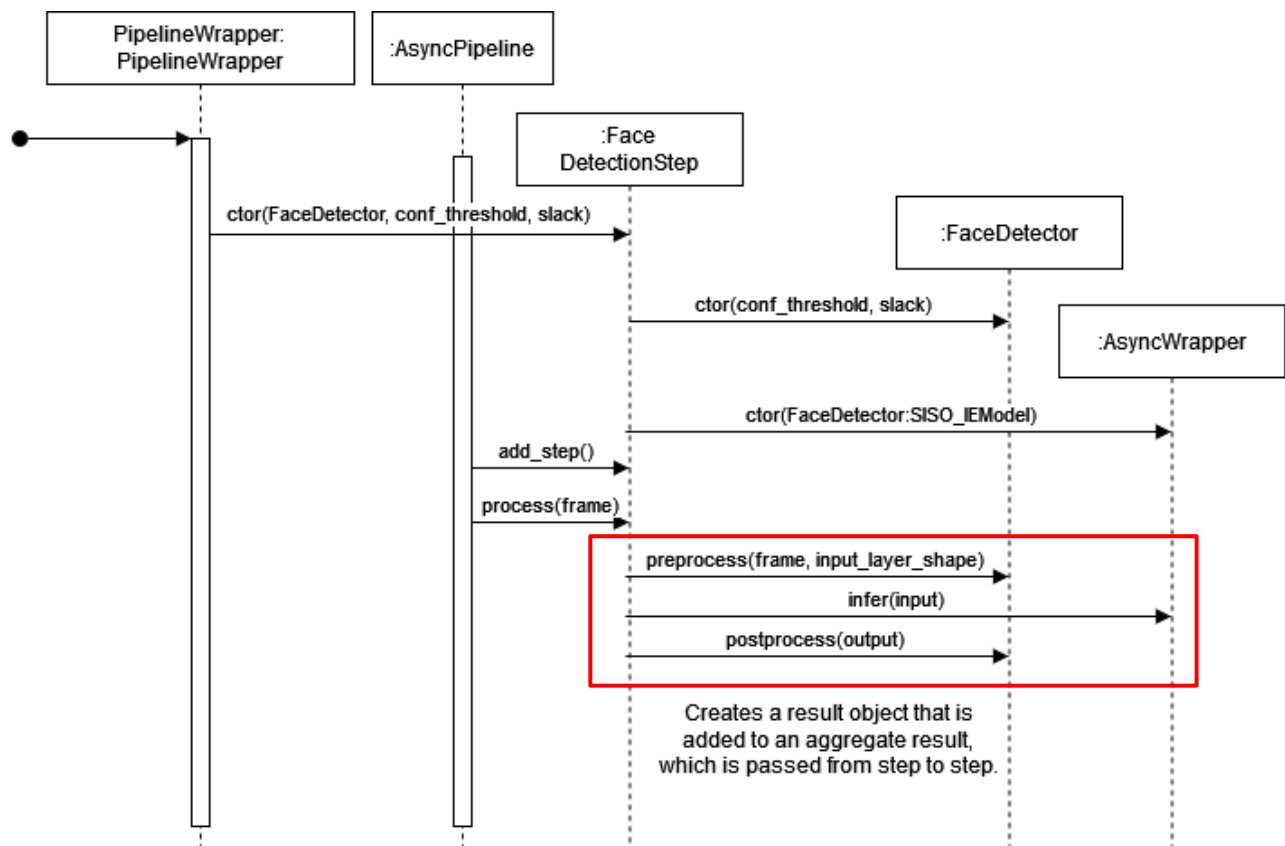


Figure 36. Steps involved in the initialization and execution of the face detection step. The highlighted executions are core to extracting the driver's face position. The same pattern is also used for the initialization and execution of the other steps.

8.2 Model Selection

8.2.1 Origin

The pipeline employs pretrained models from the OpenVINO “Open Model Zoo”, which is a collection of pretrained models by the Intel Corporation and the public specifically targeting edge use cases. This decision was primarily made to reduce development effort and thus to be able to meet project time constraints. The main benefit of using those models, apart from reduced development time, is that model architecture and tuning are specifically tailored to balance performance (FPS) and accuracy, which is crucial for a use on the edge.

8.2.2 Face Detection

The video pipeline uses the “face-detection-retail-0004”¹ pretrained model from Intel for face detection, which uses the SqueezeNet light² as a backbone to improve inference speed and usability on edge devices. The model was trained on the WIDER FACE data set by (Yang, Luo, & Tang, 2016), which consists of 393,703 labelled faces and achieves an average precision (AP) of 83%.

¹ Source: (face-detection-retail-004, 2023).

² SqueezeNet by itself is already optimized with 50x fewer parameters having a model size of less than 0.5MB as proposed by (Iandola, et al., 2016).

AP is defined as the area under the precision and recall curve, which is a standard for evaluating face detection models.

$$Precision := \frac{tp}{tp+fp}$$

$$Recall := \frac{tp}{tp+fn}$$

8.2.3 Facial Landmark Detection

The model “landmarks-regression-retail-0009”³ from Intel is used for facial landmarks detection. The model uses a classic convolution design and infers five facial landmarks, corners of the mouth, nose, and two eye centers. The eye centers are used by the pipeline to create bounding boxes required by follow-up steps.

The model was trained on the VGGFace2 data set by (Cao, Shen, Xie, Parkhi, & Zisserman, 2017) and achieved a mean normed error (NE) of 0.0705.

The mean normed error is defined as

$$\varepsilon_i := \frac{1}{N} \sum_{k=0}^{N-1} \frac{\|(\hat{\mathbf{p}}_{i,k} - \mathbf{p}_{i,k})\|_2}{d_i},$$

where N is the number of landmarks,
 $\hat{\mathbf{p}}$ and \mathbf{p} are the predicted and the ground truth respectively of the
 k th landmark of the i th sample, and
 d_i is the interocular distance of the i th sample.

8.2.4 Closed Eyes Detection

The “open-closed-eye-0001”⁴ model is used for closed eyes detection. The model is provided by Intel and uses a fully convolutional neural network to classify the eye state of an image into either “open” or “closed”.

The model was trained on the MRL Eye data set by (Fusek, 2018), which contains around 15,000 infrared images and achieved an accuracy of 95.84%.

8.2.5 Head Pose Estimation

The “head-pose-estimation-adas-0001”⁵ model is used for the head pose estimation. The model is provided by Intel and uses a convolutional neural network to estimate the yaw, pitch, and roll angles in degrees with maximum supported ranges of $[-90,90]$, $[-70,70]$, and $[-70,70]$ respectively.

The model was trained using the Biwi Kinect Head Pose database by (ETH Zuerich, 2011) and achieved a mean and standard deviation of absolute error in degrees of

- 5.4 ± 4.4 (yaw)
- 5.5 ± 5.3 (pitch)
- 4.6 ± 5.6 (roll)

³ Source: (landmarks-regression-retail-0009, 2023).

⁴ Source: (open-closed-eye-0001, 2023).

⁵ Source: (head-pose-estimation-adas-0001, 2023).

8.2.6 Gaze Estimation

The “gaze-estimation-adas-0002”⁶ model is used to estimate the gaze vector. The model is a multi-input/single-output model, where two images containing the left and right eye region and the head pose angles are used as input to produce a single gaze vector in Cartesian coordinates.

The model was trained on a proprietary data set from Intel where it achieved a mean absolute error (MAE) of the angles of 6.95 degrees and a standard deviation of the absolute error of 3.58 degrees. The MAE is defined as

$$MAE := \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

8.3 Model Deployment

Several steps are required to deploy models to the ADAS and the pipeline so that they can be used for inference (on the edge). These steps can be executed via the Jupyter notebook “models_setup.ipynb” and are required only once during system deployment. This notebook is provided as a deliverable of the project.

The names of the models are added to a models file “models.lst”. During execution of the notebook the model names are read from the file and provided to the two instances OMZ_Downloader and OMZ_Converter, which are both provided by OpenVINO.

The responsibility of the OMZ_Downloader is to check whether the required models are already available in the local cache and if not to download them from the Open Model Zoo online repository and add them to the local cache.

The responsibility of the OMZ_Converter is to use conversion information supplied along with the original model, which might have been developed using deep learning frameworks like TensorFlow 1.x, TensorFlow 2.x, PyTorch, ONNX, etc. and to convert the model to the OpenVINO intermediate representation (IR). Optimizations might be applied such as layer fusion or precision reduction. After the conversion, the model can be used for inference in the OpenVINO runtime targeting available devices such as CPU, GPU, iGPU, NCS2, etc.

Not all inference devices support the same precisions. Intel CPUs commonly support the full range, particularly floating point (FP) 16 and 32. The same applies to GPUs. The NCS2, however, can only infer using a precision of FP 16. So, if models are intended for execution on the NCS2 they need to be provided with a precision of FP 16 or converted to it.

⁶ Source: (Intel Corporation, 2023).

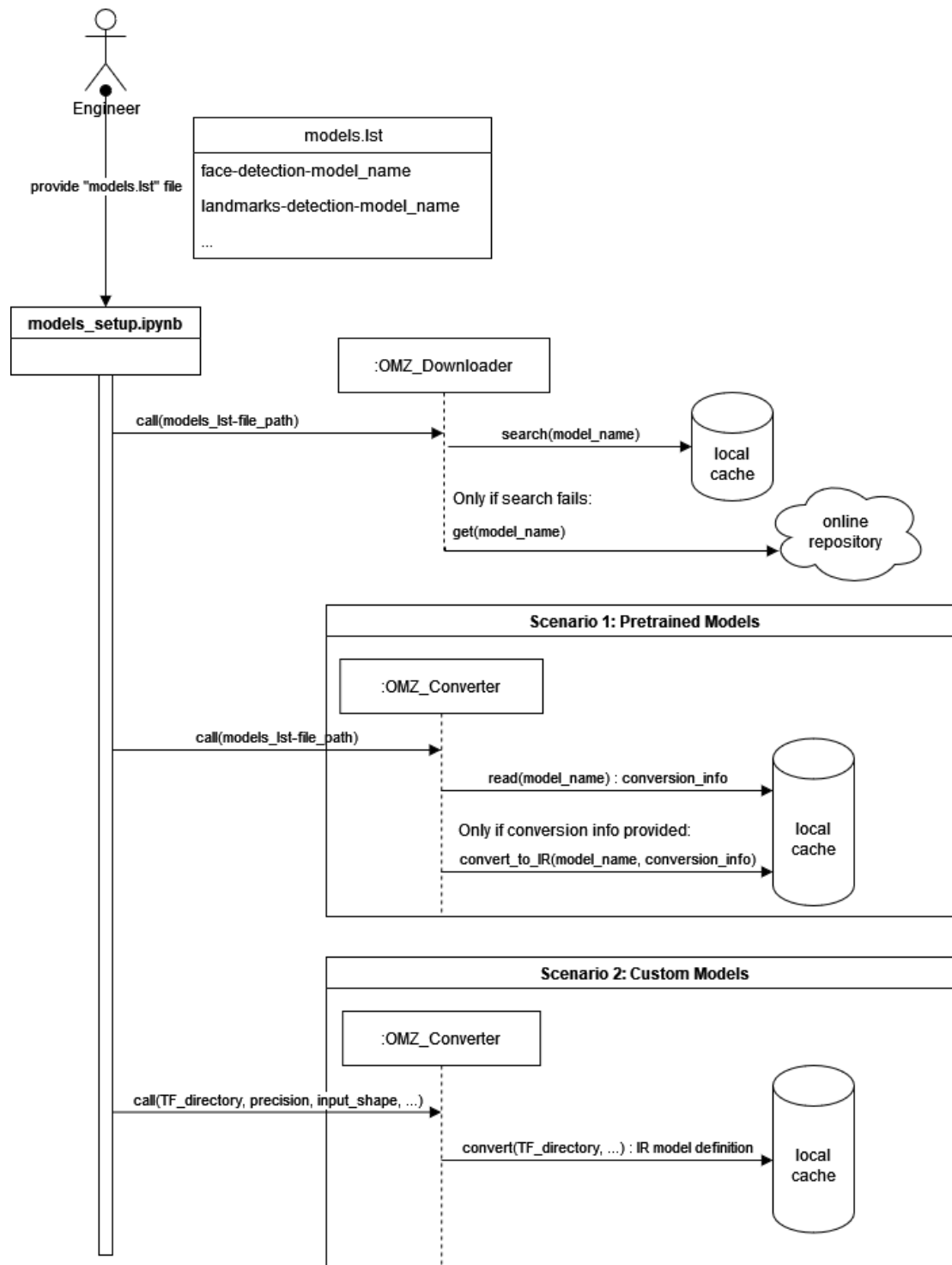


Figure 37. Process of initial model setup including model download and conversion or custom model building and conversion.

In addition to the possibility of converting pretrained models, the Jupyter notebook also provides an option to convert self-trained (custom) models that were created with the TensorFlow 2.x framework.

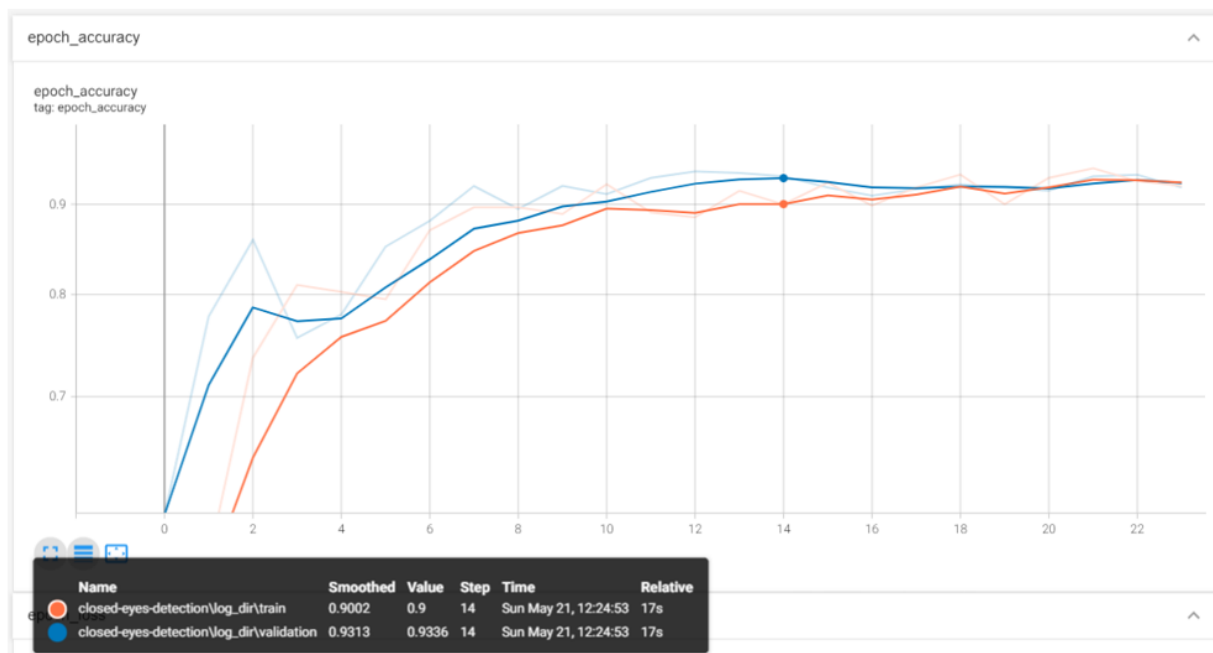
This is depicted above in “Scenario 2: Custom Models”.⁷ This notebook is provided as a deliverable of the project.

Once all models have been converted to the OpenVINO IR they are loaded into the pipeline by supplying the model definition file paths (*.xml) to the software system as command line parameters or via the launch.json during start up.

8.4 System Accuracy

8.5 Issues

The closed eyes detection of the pretrained model was insufficiently accurate, not achieving an accuracy better than a coin toss (~50%). Evaluating a similar (custom) model on a test set of 15% of the original MRL eye data set, however, showed accuracy close to the reported 95.84% of the original model. The custom model used the feature extractors of MobileNet as proposed by (Howard, Zhu, Chen, & Kalenichenko, 2017) followed by two 64 nodes dense layers. The model building Jupyter notebook is provided as a deliverable of the project.



The reason for the insufficient result on the video data, captured by the ADAS, is that the model was trained on infrared data and thus used significantly different images than those produced by the system, which uses the BGR color space.

Two attempts were made for a custom model using our own data. The data, however, was not variable enough to produce a production-ready model, i.e., the model achieved an accuracy of 98.30% on the unseen test data but produced too many false positives when used for inference on the video data. In addition, the closed eyes detection requirement could already be satisfied by the gaze estimation since closed eyes directly result in a downward gaze indicating driver distraction. Examples of this are added to a follow-up section. Therefore, and to further reduce the computational complexity of the system (in turn increasing the FPS), closed eyes detection was removed without loss of generality or effectiveness.

⁷ The verbose output of such a conversion has been added to the appendix.

9 Results

9.1 Effectiveness of the System

This section illustrates the output of the ADAS via screenshots from video streams that were annotated in real time using the NCS2 for inference of all models in the pipeline. Annotations in the frames include the face bounding box, the facial landmarks (nose, mouth corners, eye centers), the two eye bounding boxes, the head pose angles, and the estimated gaze vectors. Please note that this visual debugging aid will be disabled before going live to increase system throughput even further.

9.1.1 Gaze Estimation Results on Live Data

The following samples show that the gaze is correctly estimated for various head positions and gaze directions. Further samples, in the form of video files, are added as a deliverable of the project through the **GitHub repository**⁸. All recordings show the gaze estimation in real time.



Figure 38. Gaze estimation of a driver looking straight ahead on the left and directly into the camera on the right-hand side. The images additionally show all intermediate results of the pipeline including the bounding boxes of the driver's face, left and right eye, head pose angles, and the five landmarks indicating mouth corners, eye centers, and nose.

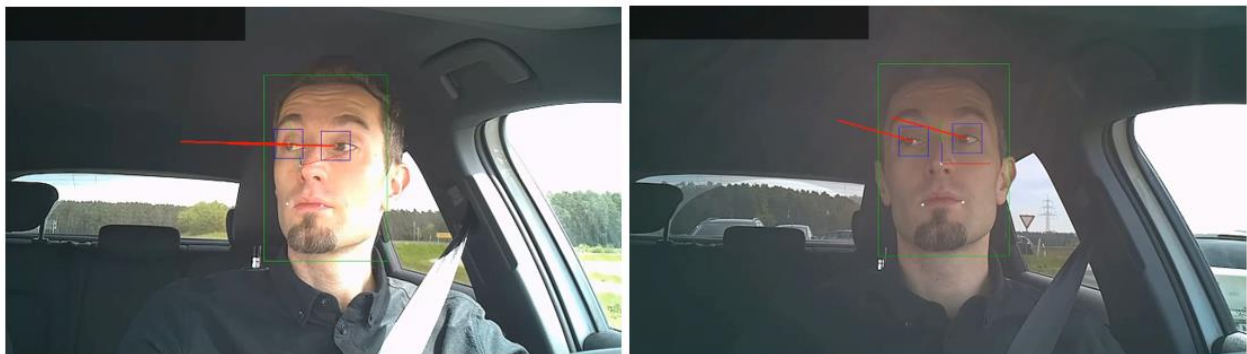


Figure 39. Gaze estimation of a driver looking to the right-hand side with different illumination illustrating the robustness under different environment conditions.

9.1.2 Distracted Driving

The system allows the specification of the allowed distraction period. Distractions exceeding this period will result in an alarm, which has been added as a warning text to the video's upper left corner for clarity.

⁸ GitHub repository URL: https://github.com/anrobr/MSDS_498-DL-Capstone_Project.

Video samples URL: https://github.com/anrobr/MSDS_498-DL-Capstone_Project/tree/master/implementation/distracted-driving-detection/output/NCS2_MYRIAD.

The allowed distraction time in all provided videos was set to 2 seconds. Please note that all recordings were done in rural areas with little to no traffic.

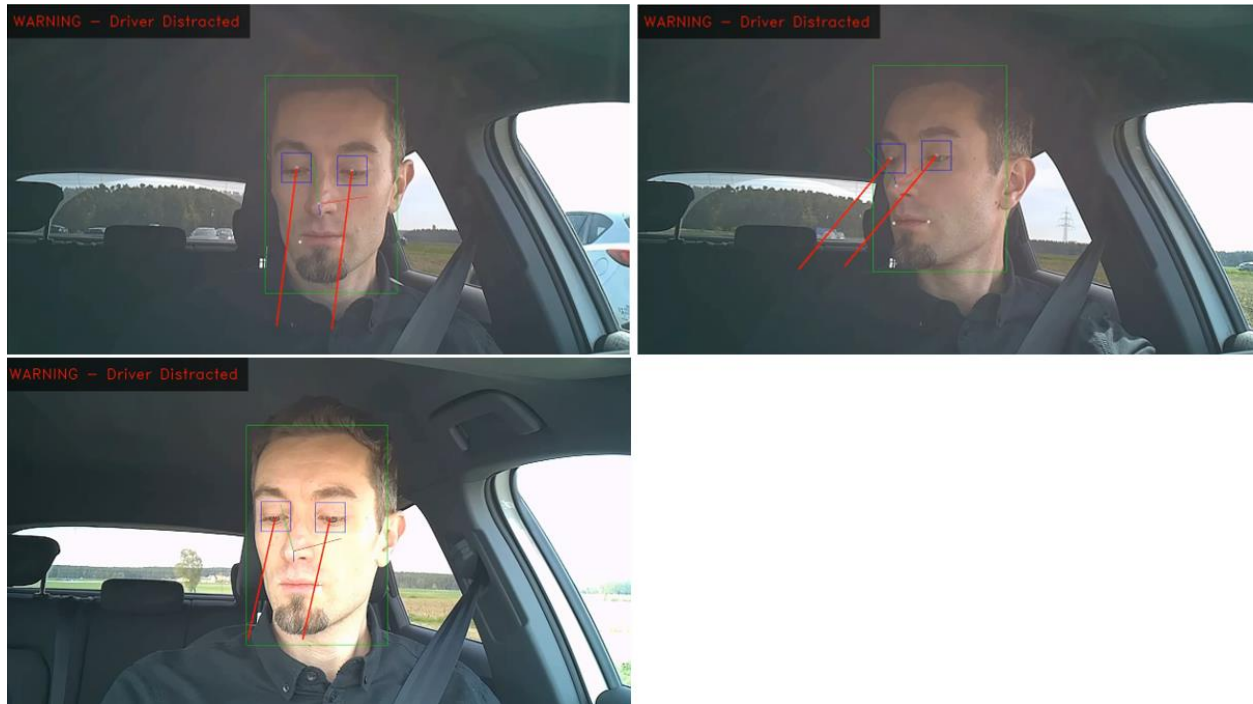


Figure 40. Gaze estimation of a driver that is distracted for a prolonged period resulting in an alarm being raised. The system operates well under varying degrees of illumination.

9.1.3 Driving with Closed Eyes

As previously mentioned, the explicit detection of closed eyes could be removed from the pipeline since this scenario was already covered by the gaze estimation. The gaze estimation results in downward vectors whenever the eyes are closed indicating a driver distraction.



Figure 41. Gaze estimation of a driver with eyes closed for a prolonged period resulting in an alarm being raised.

9.2 System Throughput

Thorough benchmarking for this project was crucial due to the real-time requirement. As previously defined the real-time requirement is met if at least 30 FPS can be processed. 30 FPS is commonly used for IVA and thus was also used for this project. All benchmarks were executed using OpenVINO API and Intel Myriad Plugin Version 2022.1.0-7019-cdb9bec7210-releases/2022/1.

9.2.1 Model Throughput

As a first step, all models were benchmarked (inference performance) separately for 60 seconds on random data. The plot below shows the maximum inference performance in FPS for the models of the pipeline per device type and precision. Maximum performance for a (sequentially executed) pipeline is the performance of the slowest pipeline step, which in this case is the face detection.

Please note that the closed eyes detection was excluded from considerations due to insufficient accuracy and its removal from the pipeline. The two other devices, namely Intel CPU (i7-13700K) and GPU (UHD Graphics 770) were added for comparison. From a pure inference standpoint, the benchmarks show that the pipeline should be fast enough to meet the real-time requirement on the edge device (NCS2) with a maximum of 119 FPS.

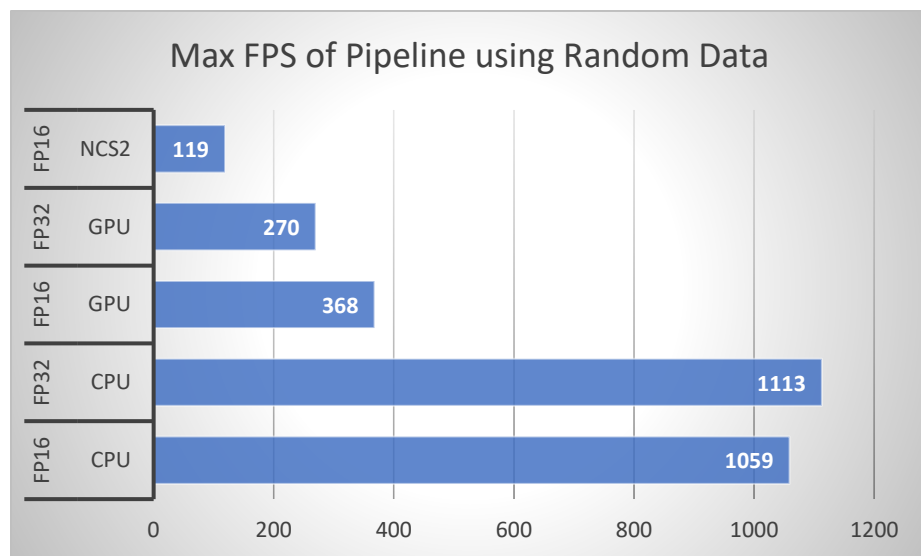


Figure 42. Max FPS of the modules needed by the pipeline per device and precision using random data.

In some cases, benchmark results from model inference on random data might significantly deviate from the inference performance on real data. To ensure that deviations, especially concerning the project's target device (NCS2) are acceptable a second benchmark was executed on real images, captured by the ADAS.

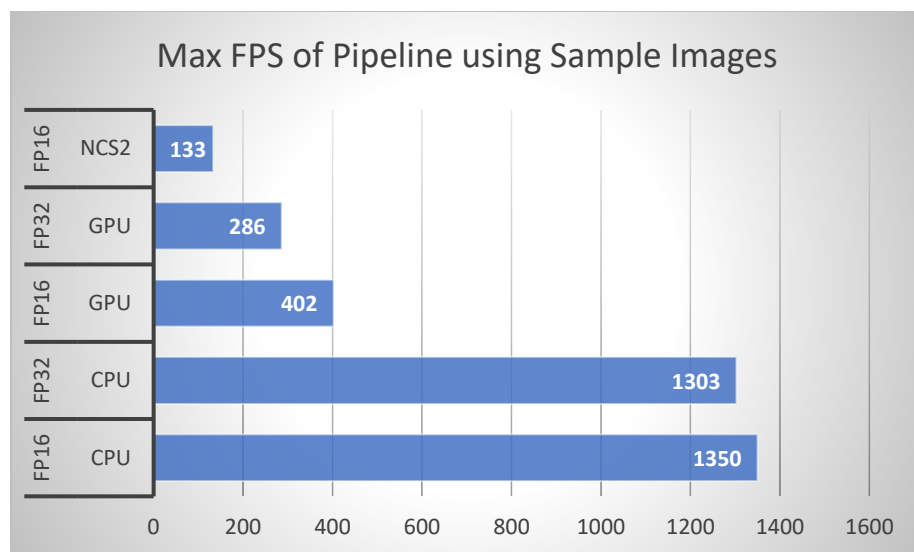


Figure 43. Max FPS of the modules needed by the pipeline per device and precision using sample images.

The inference performance of the NCS2 on sample images appears to be roughly aligned with the previous benchmark resulting in 133 FPS.

9.2.2 Pipeline Throughput

The previous model benchmarks provide only a very rough idea about the final system performance since essential pipeline steps such as video capture, pre- and post-processing, tracking, and rendering are not considered, which each adding computational overhead and thus reducing the FPS. The total pipeline performance per device and precision in FPS is shown below.

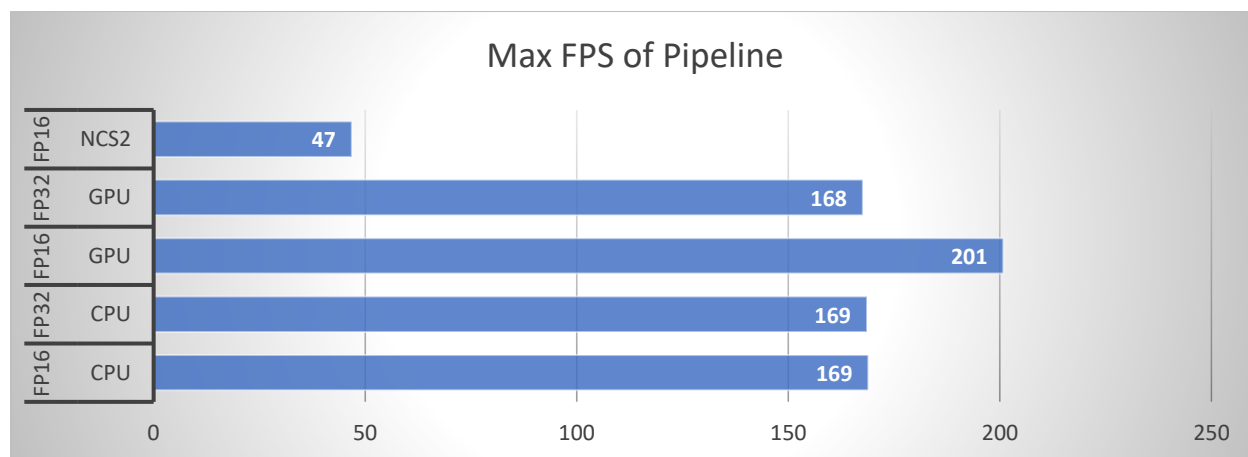


Figure 44. Max FPS of the pipeline per device and precision.

As illustrated, the intelligent video pipeline has a throughput of 47 frames per second on average and thus exceeds the real-time requirement of 30 FPS.

10 Conclusions and Outlook

This project produced an Advanced Driver Assistance System (ADAS) based on Intelligent Video Analytics (IVA) on the edge using OpenVINO and the Intel NCS2 in record time. The proposed system is not only effective in reliably detecting driver detection, the first goal of the project, but also is able to

process video data on the edge in real time, i.e., with 30 FPS. Further performance improvements are feasible, e.g., by disabling frame annotations such as bounding boxes or facial landmarks before roll-out or by employing AI component-level parallelization. So far, only gazing down based on the x-coordinates of the gaze vector is determined to be a distraction. Adding a prolonged gazing towards the left or right, however, is easy to add by considering the y-coordinates of the gaze vector as well. In addition, we propose to add a second video input that captures the driver's pose to determine poses commonly associated with distracted driving such as holding a mobile phone or smoking to enhance the detection rate further. Another valuable and easy to add extension includes taking the keeping distance into account to set the allowed distraction time dynamically.

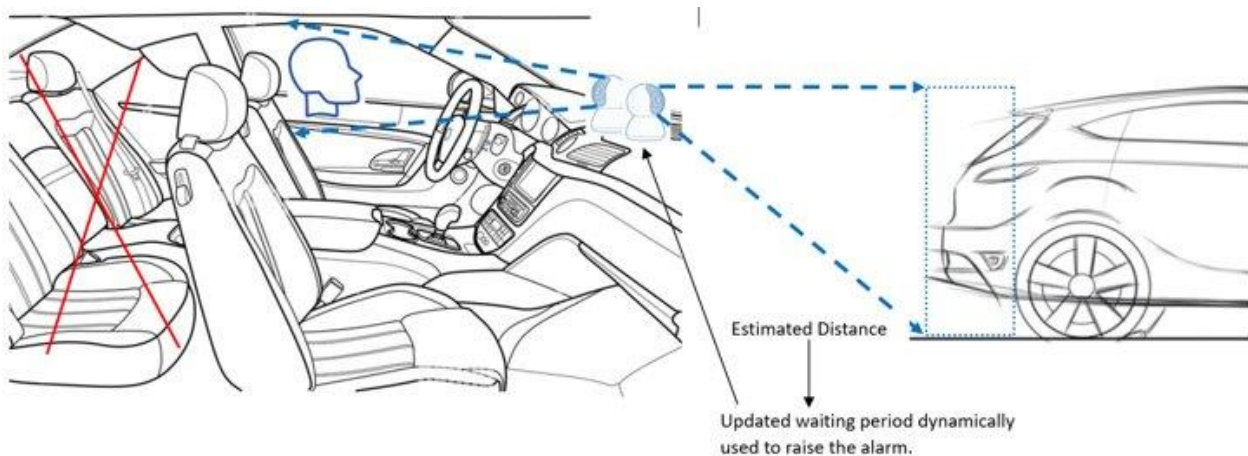
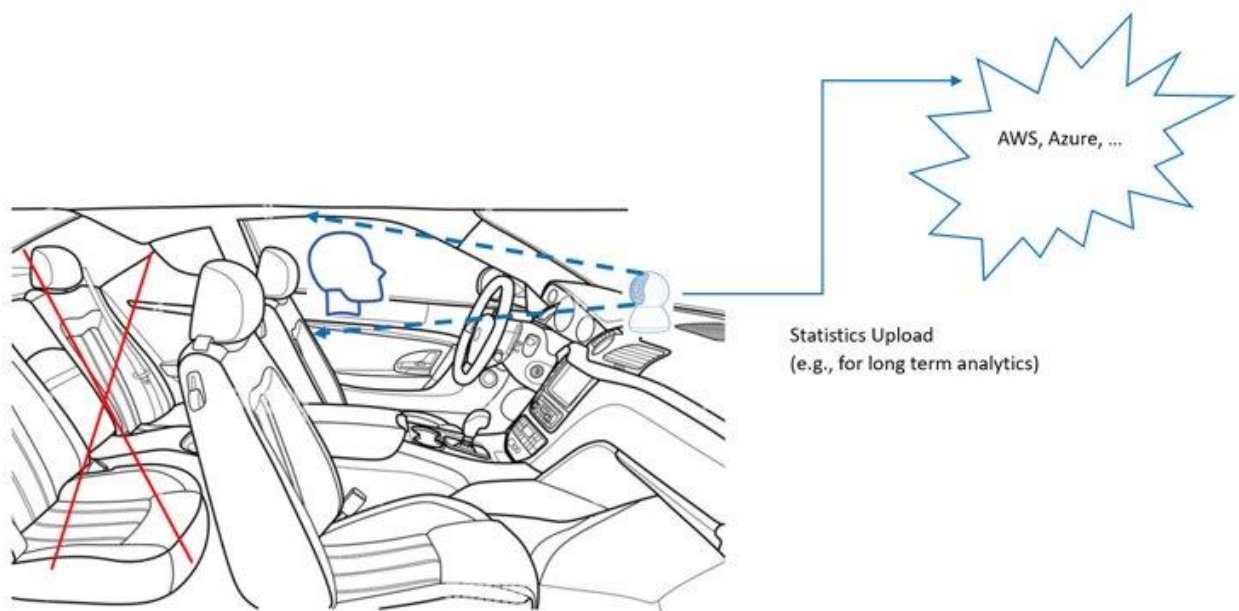


Figure 45. Depiction of an extension to the project, which uses the estimated distance to a preceding car to set the allowed distraction time dynamically.

Finally, adding cloud operability to share driver statistics to e.g., the insurance companies should help to add to the attractiveness of the ADAS.



11 References

- Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2017). *VGGFace2: A dataset for recognising faces across pose and age*. doi:10.48550/arXiv.1710.08092
- Department of Transportation, Washington, DC: NHTSA. (2021). *Traffic Safety Facts Research Note: Distracted Driving 2019*. Retrieved April 3, 2022, from National Highway Traffic Safety Administration: <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813111>
- ETH Zuerich. (2011). *Biwi Kinect Head Pose Database*. Retrieved from <https://icu.ee.ethz.ch:https://icu.ee.ethz.ch/research/datasets.html>
- Fusek, R. (2018). Pupil Localization Using Geodesic Distance. *International Symposium on Visual Computing* (pp. 433–444). Springer. doi:10.1007/978-3-030-03801-4_38
- Howard, A. G., Zhu, M., Chen, B., & Kalenichenko, D. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 1-9. Retrieved from <https://arxiv.org/pdf/1704.04861.pdf>
- Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Kreuter, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. doi:10.48550/arXiv.1602.07360
- Intel Corporation. (2021). *OpenVINO Toolkit Overview*. Retrieved from docs.openvino.ai: <https://docs.openvino.ai/2021.2/index.html>
- Intel Corporation. (2023). *face-detection-retail-004*. Retrieved from open_model_zoo: https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/face-detection-retail-0004
- Intel Corporation. (2023). *gaze-estimation-adas-0002*. Retrieved from https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/gaze-estimation-adas-0002
- Intel Corporation. (2023). *head-pose-estimation-adas-0001*. Retrieved from https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/head-pose-estimation-adas-0001
- Intel Corporation. (2023). *Intel Neural Compute Stick 2 (Intel NCS2)*. Retrieved from intel.com: <https://www.intel.com/content/www/us/en/developer/articles/tool/neural-compute-stick.html>
- Intel Corporation. (2023). *landmarks-regression-retail-0009*. Retrieved from https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/landmarks-regression-retail-0009
- Intel Corporation. (2023). *open-closed-eye-0001*. Retrieved from https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/public/open-closed-eye-0001
- Intel Corporation. (2023). *OpenVINO Home*. Retrieved from docs.openvino.ai: <https://docs.openvino.ai/latest/home.html>

- NVIDIA Corporation. (2023). *Jetson Orin Modules and Developer Kits*. Retrieved from nvidia.com: <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>
- Sinha, D. (2023, February 3). *Seamlessly Develop Vision AI Applications with NVIDIA DeepStream SDK 6.2*. Retrieved from developer.nvidia.com: <https://developer.nvidia.com/blog/seamlessly-develop-vision-ai-applications-with-nvidia-deepstream-sdk-6-2/>
- U.S. Department of Transportation. (2023). *Overview of Motor Vehicle Traffic Crashes in 2021*. Retrieved April 3, 2023, from <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813435>
- World Health Organization. (2021, October 28). *Decade of Action for Road Safety (2021-2030)*. Retrieved from who.int: <https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/decade-of-action-for-road-safety-2021-2030>
- World Health Organization. (2022, June 20). *Road traffic injuries*. Retrieved from who.int: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>
- Yang, S., Luo, P., & Tang, X. (2016). WIDER FACE: A Face Detection Benchmark. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5525-5533). <https://www.semanticscholar.org/paper/WIDER-FACE%3A-A-Face-Detection-Benchmark-Yang-Luo/52d7eb0fbc3522434c13cc247549f74bb9609c5d>: 10.1109/CVPR.2016.596.
- Zang, Kaihao, Ren, W., Luo, W., & Lai, W.-S. (2022). Deep Image Deblurring: A Survey. *International Journal of Computer Vision*, 2103-2130. doi:10.1007/s11263-022-01633-5
- Zhang, X., Sugano, Y., & Bulling, A. (2019). MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (pp. 162-175). doi:10.1109/TPAMI.2017.2778103

Appendix

A1. Verbose output of a custom TensorFlow 2.x model conversion to its OpenVINO IR

Custom model dir:

t:\498-dl\modules\project\python\gaze_estimation\python\models\custom\open-closed-eyes-002

Model Optimizer arguments:

Common parameters:

- Path to the Input Model: None
- Path for generated IR: t:\498-dl\modules\project\python\gaze_estimation\python\models\custom\open-closed-eyes-002\FP16
- IR output name: open-closed-eyes-002
- Log level: ERROR
- Batch: Not specified, inherited from the model
- Input layers: Not specified, inherited from the model
- Output layers: Not specified, inherited from the model
- Input shapes: [1,96,96,3]
- Source layout: Not specified
- Target layout: Not specified
- Layout: Not specified
- Mean values: Not specified
- Scale values: Not specified
- Scale factor: Not specified
- Precision of IR: FP16
- Enable fusing: True
- User transformations: Not specified
- Reverse input channels: False
- Enable IR generation for fixed input shape: False
- Use the transformations config file: None

Advanced parameters:

- Force the usage of legacy Frontend of Model Optimizer for model conversion into IR: False
- Force the usage of new Frontend of Model Optimizer for model conversion into IR: False

TensorFlow specific parameters:

- Input model in text protobuf format: False
- Path to model dump for TensorBoard: t:\498-dl\modules\project\python\gaze_estimation\python\models\log_dir
- List of shared libraries with TensorFlow custom layers implementation: None
- Update the configuration file with input/output node names: None
- Use configuration file used to generate the model with Object Detection API: None

```
- Use the config file:  None

OpenVINO runtime found in:  C:\Intel\openvino_env\lib\site-packages\openvino
OpenVINO runtime version:  2022.1.0-7019-cdb9bec7210-releases/2022/1
Model Optimizer version:   2022.1.0-7019-cdb9bec7210-releases/2022/1

Progress: [                ] 0.29% doneWriting an event file for the tensorboard...
Done writing an event file.

Progress: [                ] 0.57% done
Progress: [                ] 0.86% done
Progress: [                ] 1.15% done
Progress: [                ] 1.44% done
...

Progress: [.....] 99.14% done
Progress: [.....] 99.43% done
Progress: [.....] 99.71% done
Progress: [.....] 100.00% done[ SUCCESS ] Generated IR version 11 model.

[ SUCCESS ] XML file: t:\498-dl\modules\project\python\gaze_estimation\python\models\custom\open-closed-eyes-002\FP16\open-closed-eyes-002.xml

[ SUCCESS ] BIN file: t:\498-dl\modules\project\python\gaze_estimation\python\models\custom\open-closed-eyes-002\FP16\open-closed-eyes-002.bin

[ SUCCESS ] Total execution time: 7.83 seconds.

[ INFO ] The model was converted to IR v11, the latest model format that corresponds to the source DL framework input/output format. While IR v11 is backwards compatible with OpenVINO Inference Engine API v1.0, please use API v2.0 (as of 2022.1) to take advantage of the latest improvements in IR v11.

Find more information about API v2.0 and IR v11 at https://docs.openvino.ai
```