


## Article

# Implementing Computer Vision in Android Apps and Presenting the Background Technology with Mathematical Demonstrations

Roland Szabo 

Faculty of Electronics, Telecommunications and Information Technologies, Politehnica University Timisoara, Vasile Parvan Av., No. 2, 300223 Timisoara, Romania; roland.szabo@upt.ro; Tel.: +40-256-40-3351

**Abstract:** The aim of this paper is to create image-processing Android apps to launch on the Google Play Store. Three apps with different usages will be presented for different situations. The first app is a night-vision app on an Android phone that uses OpenCV. The second app is a tooth-brushing assistant application. The app is made for mobile phones and uses advanced image-processing techniques to detect when the tooth is brushed correctly or incorrectly. The main focus is on the direction of the toothbrush movement because this is one of the key aspects of correctly brushing teeth. The direction of movement of the brush is detected using movement vectors. The third app is a lane-detection app on the smartphone. Lane detection is carried out using OpenCV and TensorFlow libraries. The mobile app was implemented on the Android operating system. The app has a live video feed of the surroundings. When in the area of view, there will be a road with a lane. The system detects the lane and draws a green line over it.

**Keywords:** night vision; image motion analysis; lane detection; mobile applications; mobile computing



Academic Editor: Juan Gabriel Avina-Cervantes

Received: 9 November 2024

Revised: 3 January 2025

Accepted: 7 January 2025

Published: 9 January 2025

**Citation:** Szabo, R. Implementing Computer Vision in Android Apps and Presenting the Background Technology with Mathematical Demonstrations. *Technologies* **2025**, *13*, 27. <https://doi.org/10.3390/technologies13010027>

**Copyright:** © 2025 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

This paper focuses on creating for the Google Play Store a total of three image processing applications. The three applications incorporate image-processing techniques using OpenCV, which was compiled in the Android applications in order to do different image-processing tasks.

### 1.1. Night-Vision App

The first app is a night vision app. The night vision app optimizes your night driving by adding a unique night mode to your car's navigation system. Modifies the brightness of the screen and adjusts the color temperature to minimize eye strain and enhance visibility on nocturnal drives. This application ensures a safe and comfortable trip, allowing drivers to concentrate on the road without unwanted interference.

### 1.2. Tooth-Brushing Assistant App Using Image-Processing Techniques

The second app is a tooth brushing app. The tooth brush app transforms dental care into an engaging and efficient process by instructing users on correct brushing techniques. Through detailed tutorials and interactive elements, the app simplifies the maintenance of oral health. It is ideal for people of any age seeking to achieve comprehensive dental hygiene in a pleasant and instructive manner.

### 1.3. Lane and Car-Detection App Using Deep-Learning Algorithms with TensorFlow

The third app is a lane-detection app with deep learning algorithms. The lane-detection application enhances your driving experience by providing a virtual lane-assist feature. It utilizes real-time feedback to help drivers maintain their lane and prevent possible accidents. Developed to encourage safer driving practices, the lane-detection app is an essential tool for both daily drives and extended trips.

## 2. Related Works

### 2.1. Night-Vision App

Night vision in the automotive industry uses a thermal camera that can increase driver perception and distance of sight at night and in poor weather conditions, even in a situation where the headlights cannot increase visibility anymore [1]. These systems are installed as optional features in some expensive luxury cars [2]. The technology was first introduced in 2000 by Cadillac [3]. The technology is based on night-vision devices, which are enhanced optical devices that can improve image, thermo-imagery, and illumination [4].

This paper shows how to implement a night-vision feature on a mobile phone running Android OS [5]. With the help of mobile phone technology, many premium features of the automotive industry (navigation, GPS (Global Positioning System), HUD (Head-Up Display), night vision, and traffic-sign recognition) can be implemented for a low amount of money [6]. This affirmation is based on the fact that almost everyone has at least one mobile phone and that many people have an old, unused mobile phone that can be used for a certain embedded application [7].

By having a smartphone with many sensors and cameras, many insertion applications can be created, which can act as cheap versions of the premium features of premium vehicles [8]. Sometimes, the smartphone implementation can be used when the automotive implementation is not present in our car. Also, on mobile phones, for example, navigation apps can be better than on cars, which can have real-time traffic, danger, pit road, police location, or traffic-sign warnings. This is due to the fact that mobile phones have permanent active connectivity to the Internet, while cars do not, and there are a few people who buy a phone SIM (Subscriber Identity Module) card specially for the car to have an active connection to the Internet via 4G or 5G networks. This can only be possible when eSIM cards are built into cars in the manufacturing process.

To create a mobile phone application that can act as a vehicle night-vision system, the OpenCV library can be used. Even though mobile phones do not have thermal cameras, only if they are attached separately, using OpenCV can make an app that can mimic the night-vision feature for a luxury car. OpenCV has become advanced and has a good implementation for mobile platforms such as Android or iOS.

This paper presents the implementation of night-vision cameras for cars on Android using OpenCV.

The night system is implemented in expensive cars. If someone wants this feature on a cheaper car, it can be done with only a mobile phone and this app. No IR camera is needed. The entire night-vision system is made only with image-processing techniques using OpenCV. This is the novelty of the app. No other hardware is required, only a mobile phone. It works with any Android phone which has a rear camera. No special optics or lighting condition is needed because that is the goal of the app to make the surroundings visible in low lightning conditions. Other related published methods require an external IR camera, but the current method uses only the mobile phone's camera.

The night-vision application aims to improve visibility in low-light situations using smartphones. It uses image-processing techniques to modify brightness, contrast, and other settings to mimic night vision functions.

When comparing this application with ongoing scientific research in night vision technology, several significant differences become apparent.

The application uses digital image-processing methods on smartphone hardware, which is restricted by the quality of the camera sensor and does not have true infrared (IR) capabilities.

By contrast, contemporary night-vision technology often uses IR sensors, thermal imaging, or specialized low-light sensors. These technologies allow for the detection of heat signatures or light outside the visible spectrum, capabilities beyond what smartphone cameras can provide.

The application's effectiveness is fundamentally constrained by the smartphone's camera and ambient light conditions. It can increase visibility in low light, but it cannot function in complete darkness or in situations requiring thermal imaging.

Night-vision devices used in military, medical, or industrial fields offer far superior accuracy. These devices can work in near-total darkness and deliver precise images due to their advanced sensor technology.

This application is better suited for casual use cases, such as night photography or basic visibility improvement in low-light environments.

Night-vision technology is used in research and professional applications for critical tasks such as surveillance, wildlife monitoring, and medical diagnostics, where precision and reliability are essential.

Although the night vision application provides some functionality for casual users looking to enhance visibility in low light conditions, it is not good enough compared to the advanced capabilities of specialized night-vision devices used in scientific and professional settings.

Applications for night vision improve imaging in low-light environments utilizing sophisticated computer vision methods. Used in intelligent smart security robots on the Internet of Things, these applications incorporate event-based cameras alongside adaptive preprocessing to facilitate the identification and observation of obstacles in real time. Their primary aim is to increase visibility and decrease noise in dim conditions, making them perfect for surveillance and security tasks at night [9,10].

## *2.2. Tooth-Brushing Assistant App Using Image-Processing Techniques*

We brush our teeth every morning, before going to bed, and after meals [11]. Have we ever wondered how we might not brush our teeth correctly? Not properly brushing teeth can sometimes be worse than not brushing teeth at all [12]. We must not think that brushing our teeth involves only teeth; we must also take care of our gums [13].

It is important to know the direction of tooth brushing movements, and it is also important to know the duration of tooth brushing [14].

If we do not brush our teeth in the correct direction, it is not OK. The teeth must be brushed with the up–down movement, not with the left–right movements, because if we brush them with left–right movements, we move all the bacteria from the teeth into the gum with the toothbrush [15]. If we brush the teeth with up–down movements, then we remove the bacteria from the teeth when we reach the edge of it. The direction of brushing is not the only thing that can affect the effect of cleaning. To correctly brush the teeth, it was empirically tested that a two-minute duration is enough for good tooth brushing. In fact, each half-arc should be brushed for 30 s. A half-arc is the left or right side of the upper teeth or lower teeth. In our mouth, we have four arcs, so brushing each half arc for 30 s will be a total of two minutes of tooth brushing.

It was found that quite a few adults brush their teeth correctly; we do not talk about children learning brushing techniques [16].

Today's mobile-phone-driven society can help solve these problems to have better oral hygiene and a better life [17].

Creating an app that helps brush your teeth can be a useful tool for adults and children [18].

The tooth-brushing app can be used mainly by kids to learn to brush their teeth or by dentists to show correct tooth brushing to patients. This app has the advantage that most tooth-brushing applications require you to buy a specific (electric) toothbrush that connects to the app via Bluetooth. This app works with any standard toothbrush and relies only on image processing, with OpenCV, to recognize the movement direction of tooth brushing. This is the novelty of the app.

The tooth-brushing app aims to help users improve their tooth brushing habits. It includes features such as timers, reminders, and educational content to encourage proper brushing techniques and oral care.

The app uses timers, reminders, and possibly gamification elements to motivate users to brush their teeth for the two minutes suggested twice daily. These features are designed to increase user engagement and adherence to good oral hygiene practices.

Research indicates that behavioral interventions, such as reminders and gamification, can effectively improve oral hygiene behaviors, particularly in children. Studies have shown that applications with interactive elements can increase motivation and consistency in brushing, leading to better overall oral health outcomes.

Although the app offers tools to support better brushing habits, its success largely depends on the adherence of the user and the precision of the brushing technique, which the app cannot directly monitor or correct.

Recent studies suggest that while apps like the toothbrushing app can be advantageous, they are most effective when paired with real-time feedback systems, such as smart toothbrushes that provide immediate feedback on brushing technique (pressure, coverage, duration). This integration has been shown to significantly improve oral hygiene by ensuring that users brush properly and thoroughly.

The app is likely to provide educational content to inform users about proper brushing techniques and the importance of oral hygiene.

Educational interventions delivered through mobile apps have been proven to increase knowledge about oral health, which is crucial for preventing dental problems. However, research also highlights the need for content customization to address individual user needs and preferences, which can further amplify the effectiveness of these apps.

The tooth-brushing app offers a useful tool to promote better brushing habits through reminders, timers, and educational content. However, its effectiveness could be improved by incorporating real-time feedback mechanisms and tailored educational content, as recommended by recent scientific research. Although the app serves as a helpful addition to oral hygiene practices, the incorporation of smart technology could markedly enhance its impact on users' oral health.

Tooth-brushing applications use computer-vision technology to improve dental hygiene by monitoring user movements and providing instant feedback. These applications utilize facial recognition and augmented reality (AR) overlays to ensure full brushing coverage. Using pose-estimation models, they guide users, thus improving dental care habits and making the process more engaging, especially for children [9,10].

### *2.3. Lane- and Car-Detection App Using Deep-Learning Algorithms with TensorFlow*

In today's world, self-driving cars and lane-detection cars have started to emerge in our lives faster than we can imagine [19].

There is no need to talk about robots [20], which are only in the prototype state [21], but we can talk about robots that are present in our daily life and seem to be natural [22]. Robots like this are drones, robot vacuum cleaners, mops, window cleaners, and, of course, cars with some level of autonomous driving [23].

For autonomous driving, more sensors can be used [24], such as LiDARs or video cameras [25].

Video cameras can be a good approach [26] due to the existence of the vast OpenCV image-processing library [27].

One autonomous driving feature for cars is the detection of the lane [28]. For a car to be able to drive autonomously [29], having the lane-detection function installed is a good feature [30]. The lane-detection function can also help drivers keep the car on the road [31] when tired after a long drive [32].

Lane detection can be performed with image-processing techniques [33], but in addition to this, much more is needed, such as a trained neural network, to correctly detect the lane of the road and exclude false positives [34].

Lane detection and car detection are present in expensive cars. If a user wants this feature on a cheaper car, it can be done with only a mobile phone and this app. All lane detection and car detection are done with image processing, OpenCV, and the usage of a trained neural network using TensorFlow. This is the novelty of the app.

The lane-detection application is created to identify lanes, mainly using the smartphone camera to help drivers remain within their lanes. It probably employs image-processing algorithms to recognize and follow the road lane markings.

The application uses simple computer-vision techniques to detect lane lines from the video feed captured by the smartphone camera. This might involve edge-detection algorithms such as Canny edge detection [35], or the Hough transform [36] to identify straight or curved lines representing lane boundaries.

Modern lane-detection systems, particularly those utilized in advanced driver-assist systems (ADAS), use more advanced technologies. These include deep-learning models such as Convolutional Neural Networks (CNNs), trained on extensive datasets to accurately recognize and forecast lane positions under various conditions such as poor lighting or road obstacles. These models also incorporate data from multiple sensors (e.g., LiDAR [37], radar) to increase robustness.

The precision of lane detection depends on the quality of the smartphone camera and the environmental conditions (e.g., lighting, visibility of the road markings). Since it relies solely on the camera, its performance can be hindered by bad weather or if lane markings are worn or blocked.

Current research indicates that high-accuracy lane detection requires a combination of high-resolution cameras, sensor fusion, and advanced algorithms capable of handling different road conditions. Systems like Tesla's Autopilot [38] or Waymo's self-driving technology [39] use multiple sensors and redundancy to guarantee high reliability, which is crucial for safety in autonomous driving.

Considering that the application operates on a smartphone, it is probably constrained by the processing power of the device. Real-time lane detection might experience latency, especially on older devices or during complex image-processing tasks. The app can run on any standard Android phone. The minimal platform supported by OpenCV Java API is Android 2.2 (Froyo or API 8). Presently, almost all Android devices run a newer Android version than Android 2.2 Froyo. The latest Android version is Android 15, and all users are encouraged to run the latest Android version, if possible. The idea is that the better the phone hardware, the better will run the image processing on that mobile phone.

Cutting-edge lane-detection systems are intended for real-time processing with minimal delay. They employ dedicated hardware accelerators (such as GPUs or specialized AI chips) to ensure that lane detection and response times adhere to the strict requirements of high-speed driving.

This application might be appropriate for general driving assistance, providing an extra layer of safety by alerting drivers when they drift out of their lane.

Conversely, lane-detection systems in research and industry are essential components of fully autonomous vehicles, where they are integrated with other safety and navigation systems to deliver comprehensive driving solutions.

The lane-detection application offers a valuable tool for basic lane detection on a smartphone, suitable for casual driving assistance. However, compared to the sophisticated systems developed in scientific research and used in the automotive industry, it is not accurate, reliable, or capable of performing in real time. Advanced lane-detection systems use more intricate algorithms and sensor integration, delivering significantly better performance under a variety of driving conditions.

Applications for lane detection play an essential role in driving assistance systems. They use deep-learning methodologies, such as convolutional neural networks (CNNs), to identify road lanes in real time in diverse environmental scenarios. These applications improve navigation safety by offering precise lane tracking and supporting technologies for autonomous driving [9,10].

Table 1 shows a comparison of the state of the art in computer-vision mobile apps [9,10].

**Table 1.** Summary table comparison of the state of the art in computer-vision mobile apps [9,10].

App	Primary Application	Key Strategies	Advantages
Night Vision	Surveillance in low-light conditions	Adaptive preprocessing combined with event-based cameras	Enhanced clarity and diminished noise
Tooth Brushing	Observation of oral hygiene	Face identification, augmented reality, posture estimation	Improved brushing practices
Lane Detection	Assistance with driving	CNNs in deep learning	Real-time lane tracking with precision

### 3. Problem Formulation

#### 3.1. Night-Vision App

Driving at night frequently requires managing the intense glare of screens and bright displays, which can cause eye strain and reduce driving concentration. Numerous navigation apps currently do not have adaptive night mode capabilities, which are necessary to provide a safer and more comfortable experience during night travel.

The project involved developing a night vision app for Android smartphones using OpenCV.

Smartphones do not come with thermal cameras unless they are added as external peripherals.

Due to the sophisticated image processing and filtering methods in OpenCV, it is feasible to achieve results similar to those produced by night-vision systems in high-end vehicles.

The outcome would be nearly identical, but the cost would be significantly lower, bordering on negligible.

### 3.2. Tooth-Brushing Assistant App Using Image-Processing Techniques

Several people fail to brush their teeth properly, resulting in inadequate oral hygiene and preventable dental issues such as cavities and periodontal disease. There is an absence of captivating and easy-to-use resources to instruct individuals on proper brushing methods and to make the experience more enjoyable.

The project involved developing a mobile app to assist individuals in brushing their teeth properly.

This involved identifying the toothbrush's motion direction and implementing a brushing timer.

To detect the brushing direction, image processing with OpenCV should be employed. Moreover, a brushing timer feature must be integrated. The timer aids users in knowing the remaining brushing time, ensuring they brush for at least two min.

### 3.3. Lane and Car-Detection App Using Deep-Learning Algorithms with TensorFlow

Unintentional lane changes and lane drifting are major factors in road accidents. Drivers frequently do not have access to real-time tools that offer prompt lane assistance and guidance, which can aid in maintaining safer driving practices and avoiding crashes.

The objective was to develop a lane-detection application. The decision to build the app for mobile phones was made to enhance its portability and simplicity of use in vehicles.

Android was selected as the operating system due to its status as the most extensively used mobile platform, featuring integrated image-processing libraries.

## 4. Problem Solving

### 4.1. Night-Vision App

An app was created that has a night-vision function using only OpenCV image-processing techniques.

This app can be used as a night-vision app for cars, but it can also be used as an app that is useful for searching for a lost object at night when light torches cannot help.

In automotive usage, it can be used to avoid collisions with pedestrians or animals.

The smartphone must be mounted on the windshield with a cell phone stand for vehicles. It must be ensured that the back camera of the smartphone faces the road ahead.

An old smartphone can be used with the night-vision app.

The application enjoys the benefit of showing just the street in night vision, with no different signs, which can sometimes be too much. Ideally, this application will help drivers avoid accidents with humans or animals crossing the road.

It is a good habit to use a car phone charger, as the screen of the smartphone is always on and can rapidly deplete the battery of the smartphone.

The night-vision app does not use near-infrared cameras used for ADAS in the automotive industry, but it can be quite effective when we do not have this premium feature in our car. The night-vision app is just an enhancement algorithm, but it can be used when someone does not have the night-vision feature in their car. The novelty lies in the fact that there is no such app as this in the Google Play Store.

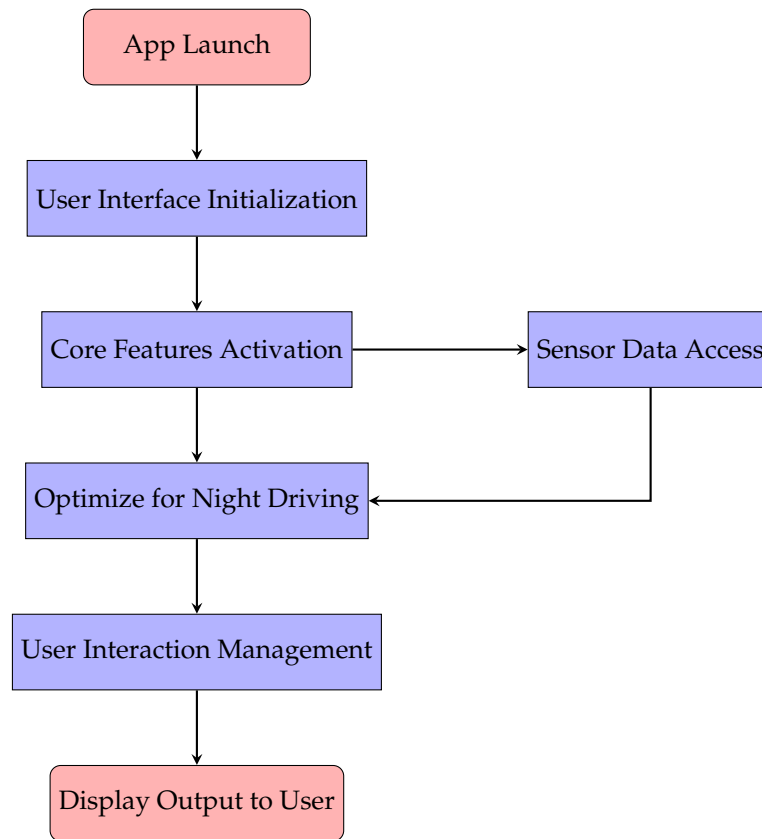
Figure 1 shows the architecture diagram of the night-vision app.

Equations (1)–(4) were used to create the night-vision app.

Using  $x$  and  $y$  as spatial coordinates, let us represent the input image as  $I(x, y)$ .

The function  $\text{Enhance}(I(x, y))$  can be used to model the image-enhancement process, where the visibility of the input image is improved.

$$\text{Enhanced\_Image}(x, y) = \text{Enhance}(I(x, y)) \quad (1)$$



**Figure 1.** The architecture diagram of the night-vision app.

To see better at night, night vision (NV) frequently involves increasing low light levels. This amplification procedure is shown below.

$$NV(I(x, y)) = k \cdot I(x, y) \quad (2)$$

where NV is the night-vision function and the constant  $k$  stands for the amplification factor. Combining the effect of night vision with image enhancement:

$$\text{Enhanced\_NV}(x, y) = \text{Enhance}(NV(I(x, y))) \quad (3)$$

This formula depicts the improved image after the night-vision effect was included.

Figure 2 shows the real image and the enhanced software image made with the mobile-phone app.

To further increase visibility, thresholding can be added to the improved image. We shall use  $T$  to represent the threshold value.

$$\text{Thresholded\_Image}(x, y) = \begin{cases} \text{Enhanced\_NV}(x, y), & \text{if } \text{Enhanced\_NV}(x, y) > T \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

To further improve visibility, the pixel values below a threshold  $T$  are set to zero using a thresholding procedure represented by this formula.

The following images were tested in really low visibility to see that the night-vision app works. Some of them might be blurry, but compared to the low-visibility conditions, they can be used to see some objects at night.





**Figure 2.** The image viewed by the mobile-phone app overlaid on the real image with poor visibility.

Figure 3 shows the actual screenshot from the mobile app while looking at a street at night.



**Figure 3.** A screenshot that was viewing the poor-visibility street at night from the previous figure.

Figure 4 shows the Android night-vision app.

The test was carried out under harsh conditions to test the app to the maximum. It was night and it was snowing, so there was close to no visibility, but the app can still see well at night and in poor weather conditions. It can be seen that at night, due to snowy weather, visibility is low, if not close to zero. In addition, the windshield shows a blurred image due to the moisture obtained by the melted snow that fell on the warm window.



**Figure 4.** A screenshot made with the mobile-phone app, which was viewing the street at night.

Even in these conditions, the software tries to make the images show well in these poor lighting and weather conditions. In the real image, only the first car that is the closest can be seen, but no buildings or other cars are seen in the background.

In the enhanced mobile-phone app, almost everything looks like daylight. Only a small noise is overlaid on the image viewed.

The software enhancement steps are the following: convert the image to grayscale, make a histogram equalization, and then make an adaptive gamma correction using a lookup table.

In case the weather is better and the only problem is that there are low lighting conditions, the app works even better, and everything can be seen more clearly.

The clarity of the viewed image can be seen, which was made at night when it was snowing. Not only the buildings in the background but also the people walking on the street can be seen.

It is clear how well the filtering was carried out. In some night-vision apps, when in the viewed image there is a light source, due to histogram equalization, the image will be overlaid by a white spot. In the current mobile-phone app, there is a light source, but the white spot is reduced to a minimum.

#### *4.2. Tooth-Brushing Assistant App Using Image-Processing Techniques*

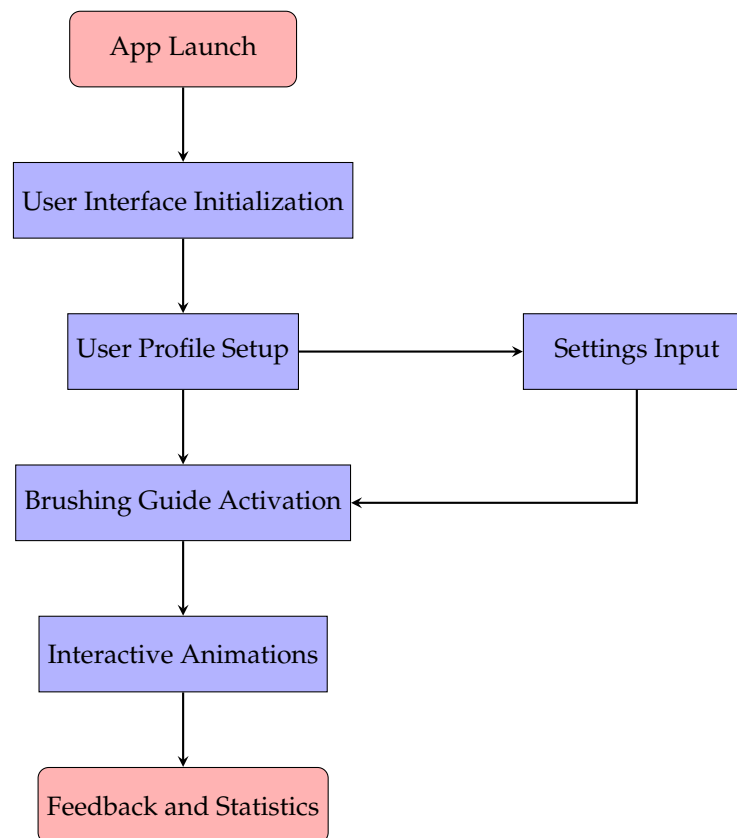
The app was implemented for the Android operating system for mobile phones.

Android is the most widely distributed operating system for mobile phones.

The mobile-phone implementation is the most convenient because almost everyone owns a mobile phone, and users are excited to use mobile-phone apps for different tasks. Mobile phones make our lives easier.

Figure 5 shows the architecture diagram of the tooth-brushing app.

There is an advantage in using the mobile-phone environment because it has specially built OpenCV image-processing libraries for mobile operating systems (Android and iOS). The implementation of this paper will present an Android app built for Android. The only thing a mobile phone needs is a front camera, which is present in almost all modern phones. Mobile phones were chosen because tablets are also good for use while brushing teeth, but the application will also work quite well on tablets.



**Figure 5.** The architecture diagram of the tooth-brushing app.

To detect the direction of movement in brushing, two consecutive frames are used. The pixels in two consecutive frames are united.

To ease the system, not all pixels are united, but only certain ones. Initially, there is a grid. In addition, only pixels from the grid are united. The grid can be made with larger or smaller resolutions, but there is no need to unite all the pixels for movement detection. If there is no movement, the movement vector will be a dot; if there is movement, the movement vector will show the direction of the movement. After this, all it must be done is to detect if the direction is up–down or left–right. This can be detected easily by analyzing the coordinates of the points of moment vectors. After this, a decision can be made if the movement of the toothbrush is incorrect (left–right) or correct (up–down).

All that needs to be done is to overlay a timer on the image, allowing the user to know the duration of the brushing.

The pause of the stop buttons must be added to control the brushing timer.

The relative motion of the front camera does not influence the quality of detection of tooth brushing. The novelty lies in the fact that the app uses movement vectors to detect toothbrush movements and in the fact that there is no such app like this on the Google Play Store.

The following formulas were used for the toothbrush app.

The movement of a characteristic between consecutive frames is represented by the displacement vector represented in Equation (5).

$$\vec{v} = (dx, dy) \quad (5)$$

The horizontal and vertical displacements are denoted by  $dx$  and  $dy$ , respectively, in this instance.

The velocity vector shows how quickly a feature's position changes over time, represented in Equation (6).

$$\vec{v} = \left( \frac{dx}{dt}, \frac{dy}{dt} \right) \quad (6)$$

The horizontal and vertical components of the velocity are indicated, respectively, by  $\frac{dx}{dt}$  and  $\frac{dy}{dt}$ .

The apparent motion of objects in an image due to the relative motion of the observer and the scene is represented by the optical flow represented in Equation 7.

$$\vec{F} = (u, v) \quad (7)$$

The horizontal and vertical components of optical flow are denoted by  $u$  and  $v$ , respectively, in this instance.

Motion estimate is a technique used to facilitate tasks such as object tracking and video reduction by estimating the movement of objects between frames represented in Equation (8).

$$\vec{M} = (dx, dy, dt) \quad (8)$$

In this case, the temporal displacement is represented by  $dt$ , and the spatial displacement is represented by  $dx$  and  $dy$ .

The motion of the entire image as a result of the scene or the motion of the camera is represented by global motion vectors represented in Equation (9).

$$\vec{G} = (tx, ty) \quad (9)$$

In this case, the translational motion in the horizontal and vertical directions is denoted by  $tx$  and  $ty$ , respectively.

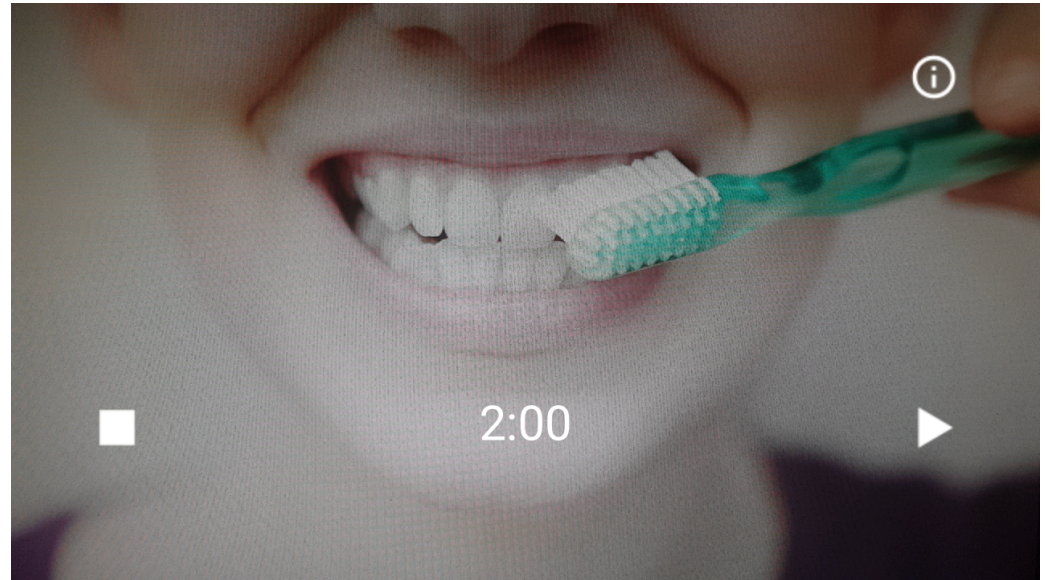
Figure 6 shows the initial screen of the app. It can be seen that the timer can count down from two minutes to allow the user to wash their teeth for two minutes. The play and stop buttons, which can start or stop the timer and detect the direction of tooth brushing, can also be seen. The information button is also available, which will show a small hint on how to use the app.

Figure 7 shows that the tooth-brushing app is started after pressing the play button. The timer starts to count down, and the direction detection is also started. The movement vectors of the app can be seen, which are drawn over the live image feed. The pixels of two consecutive frames are united with the movement vectors. If there is no movement, the vector will be shown as a dot, and if there is movement, then the direction of the movement will be shown as a line, with the movement vectors, by uniting the initial and current positions of the moving pixel. If the vector is vertical, then there is vertical movement (up–down) of the toothbrush, which is correct. If the vector is horizontal (left–right), then there is a horizontal movement of the toothbrush, which is incorrect. The standard tooth-brushing technique on many dentists' websites is with small up–down circular movements, not big left–right linear movements, and this is what the app tests with image processing. However, in the actual brushing process, such as the Bass brushing technique, horizontal brushing is required when cleaning the occlusal surfaces of teeth. The movement vectors are shown on the whole image, not only on the teeth. The movement vectors do not have to be drawn only on the teeth; they show the direction of the toothbrush movement, mainly left–right or up–down; the correct movement is up–down, and this is what the app recognizes. In the image, the pause and stop buttons can also be seen. The stop button will reset the timer to two minutes. The pause button will stop brushing your teeth, and

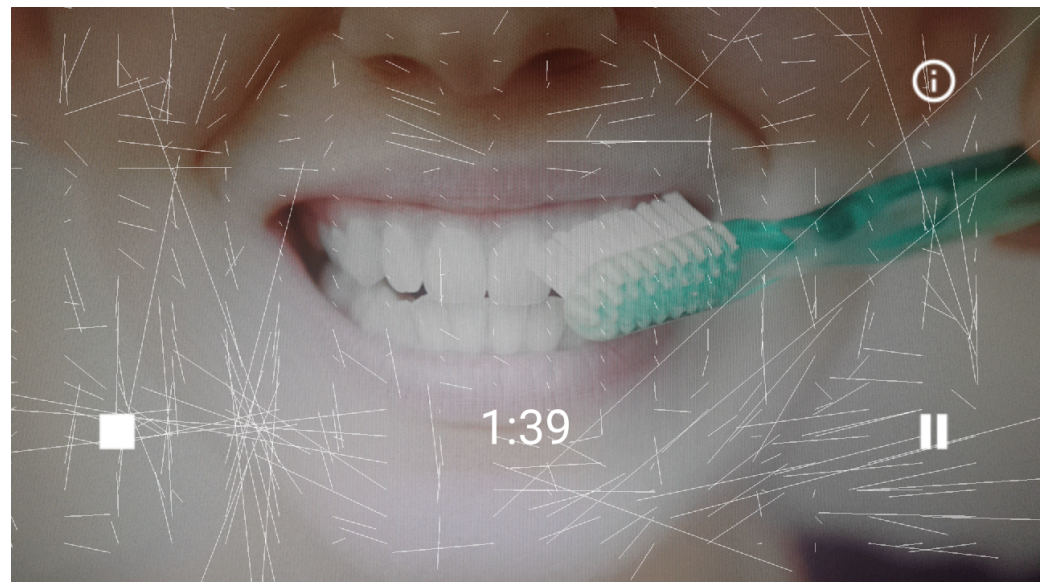
the timer will not be reset. The pause button stops the detection of the movement of the toothbrush.

When the tooth brushing is paused, the movement vector is not drawn in the live video feed captured during tooth brushing.

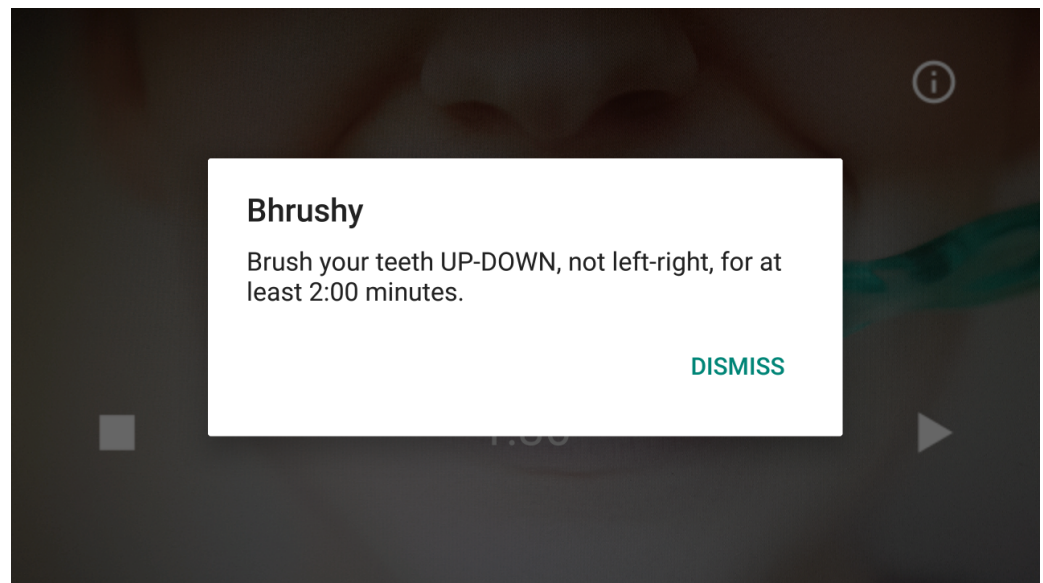
Figure 8 shows the help screen of the tooth-brushing app. There is information on how to use the app and how to brush your teeth correctly. The teeth must be brushed with up–down movements, not left–right movements, for at least two minutes.



**Figure 6.** Initial screen of the tooth-brushing app.



**Figure 7.** Tooth-brushing app with movement vectors.



**Figure 8.** Help screen of the tooth-brushing app.

#### 4.3. Lane and Car-Detection App Using Deep-Learning Algorithms with TensorFlow

The lane-detection app uses a TensorFlow database-trained neural network to recognize road lanes and cars. There is no such app on the Google Play Store.

The following formulas were used in order to implement lane detection.

The first relationship used is presented in Equation (10).

$$A_{cc} = \frac{\sum clip C_{clip}}{\sum clip S_{clip}} \quad (10)$$

where  $A_{cc}$  is the precision, in  $clip$   $C_{clip}$  are the number of estimated lane points that are properly cared for, and  $S_{clip}$  are the number of ground-truth points.

Figure 9 shows the architecture diagram of the lane-detection app.

The  $F_1$  (Precision and Recall function) can also be computed as follows using Equation (11).

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

where precision is presented in Equation (12).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

The Recall is presented in Equation (13).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (13)$$

where TP is the total points, FP is the false positives, and FN is the false negatives [40].

To detect lanes in photos using TensorFlow, a neural network model must first be trained to predict lanes. It will be given a mathematical illustration of a Convolutional Neural Network (CNN) architecture using TensorFlow for lane detection.

$$\mathbf{Z}^{[l]} = \text{Conv\_2D}(\mathbf{A}^{[l-1]}) \quad (14)$$

Equation (14) is the result of applying a convolutional layer  $l$  to the activation  $\mathbf{A}^{[l-1]}$  of the previous layer. The input image is subjected to a convolutional filter through the Conv\_2D operation.

$$\mathbf{A}^{[l]} = \text{ReLU}(\mathbf{Z}^{[l]}) \quad (15)$$

Equation (15) is the activation  $\mathbf{A}^{[l]}$  of layer  $l$ , where the convolutional layer output  $\mathbf{Z}^{[l]}$  is subjected element-wise to the rectified linear unit (ReLU) activation function.

$$\mathbf{A}^{[l]} = \text{MaxPooling\_2D}(\mathbf{Z}^{[l]}) \quad (16)$$

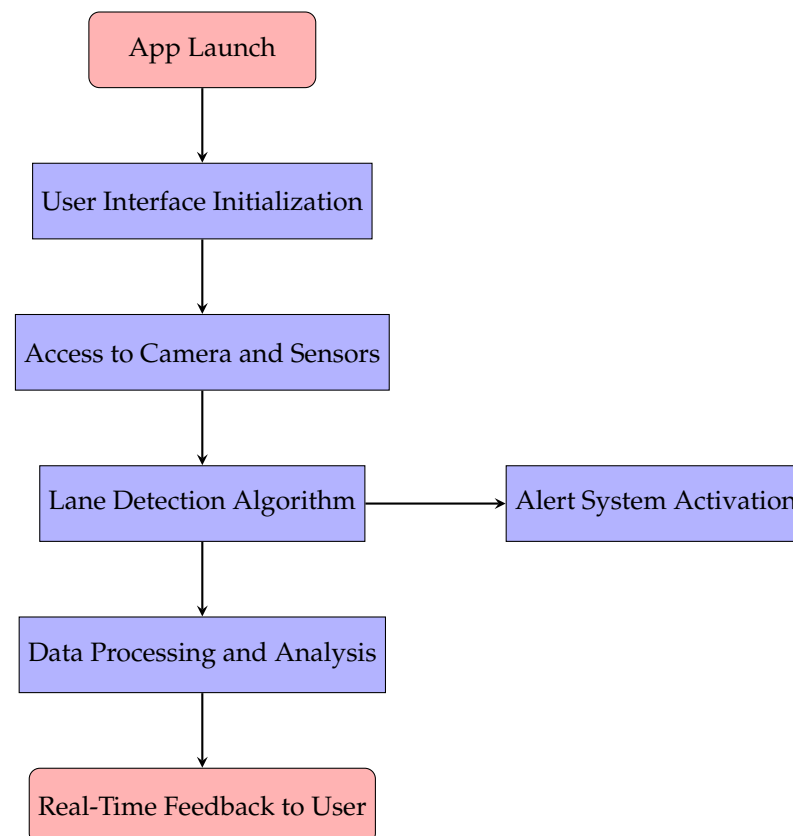
Equation (16) is the result of applying a max pooling layer to the output  $\mathbf{Z}^{[l]}$  of the preceding layer, representing the output  $\mathbf{A}^{[l]}$ . Max pooling preserves significant characteristics while reducing the spatial dimensions of the feature maps.

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \quad (17)$$

Equation (17) is a representation of the output of the fully connected layer  $\mathbf{Z}^{[l]}$ , where  $\mathbf{b}^{[l]}$  is the bias vector,  $\mathbf{A}^{[l-1]}$  is the activation of the layer before it and  $\mathbf{W}^{[l]}$  are the weights.

$$\hat{y} = \text{Softmax}(\mathbf{Z}^{[L]}) \quad (18)$$

Equation (18) addresses the result  $\hat{y}$  of the last layer of the network, where the Softmax function is applied to the result  $\mathbf{Z}^{[L]}$  of the last completely associated layer. The Softmax function yields a likelihood distribution over the classes (for example, the presence of lane markings).



**Figure 9.** The architecture diagram of the lane-detection app.

These mathematical expressions offer a high-level summary of a common CNN architecture for lane detection based on TensorFlow. In practice, this involves defining the architecture, assembling the model, training it using annotated image data, and assessing its output.

Next, loss function and error plots used in training and testing the neural network will be presented.

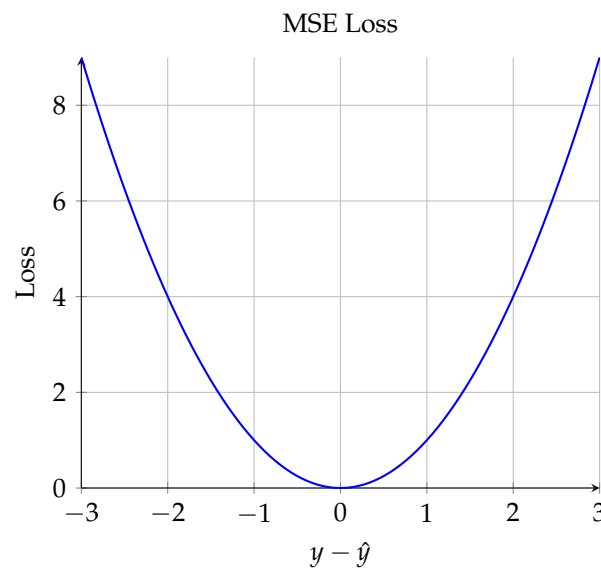
Mean Squared Error (MSE) loss calculates the mean of the squared differences between the predicted values and the actual values as shown in Equation (19).

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (19)$$

where:

- $n$  represents the total count of samples,
- $y_i$  is the actual value,
- $\hat{y}_i$  denotes the predicted value.

The MSE loss plot is presented in Figure 10.



**Figure 10.** MSE loss.

Cross-Entropy Loss is frequently applied to classification problems. It quantifies the disparity between two probability distributions, the true labels, and the predicted probabilities.

The cross-entropy loss for binary classification is outlined in Equation (20).

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (20)$$

where:

- $n$  denotes the total sample count,
- $y_i$  represents the binary true label,
- $\hat{y}_i$  is the estimated likelihood of belonging to the positive class.

The binary cross-entropy loss plot is presented in Figure 11.

In multi-class classification, the following section introduces the categorical cross-entropy loss as shown in Equation (21).

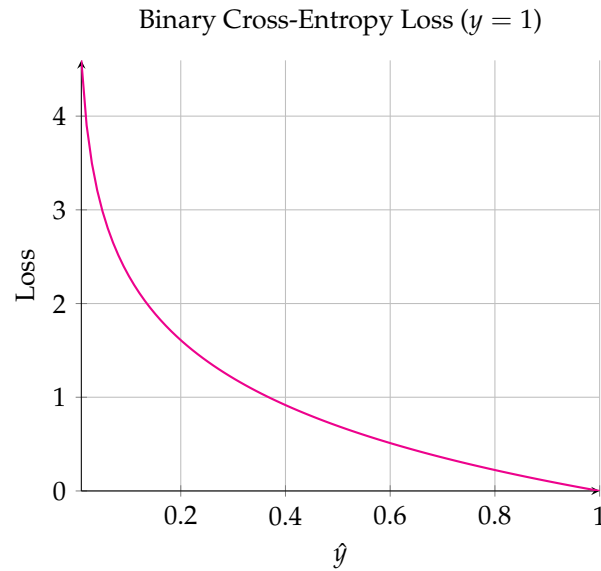
$$\mathcal{L}_{\text{CCE}} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k y_{ij} \log(\hat{y}_{ij}) \quad (21)$$

where:

- $n$  represents the total number of samples,

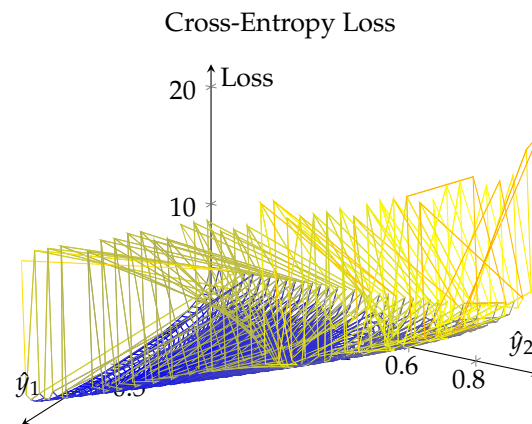


- $k$  denotes the count of classes,
- $y_{ij}$  is a binary indicator (0 or 1) indicating whether the class label  $j$  correctly classifies sample  $i$ ,
- $\hat{y}_{ij}$  is the forecasted likelihood that sample  $i$  belongs to class  $j$ .



**Figure 11.** Binary cross-entropy loss.

The multi-class classification, the categorical cross-entropy loss plot, is presented in Figure 12.

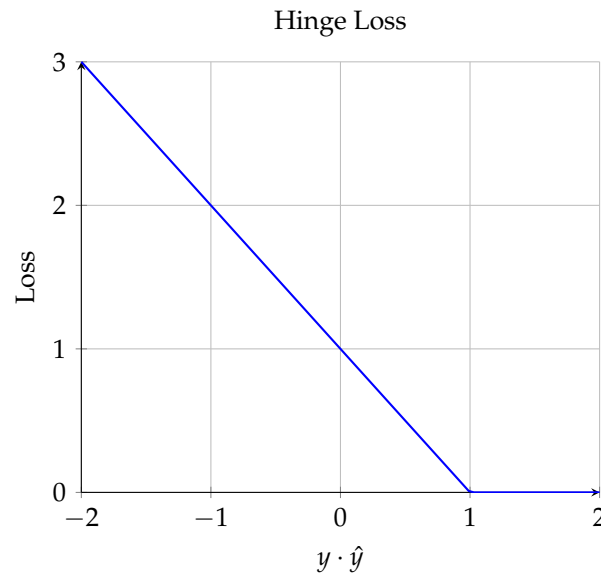


**Figure 12.** Multi-class classification cross-entropy loss.

Function: The cross-entropy loss for an actual label  $y = [1, 0, 0]$  and predicted probabilities  $\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3]$  (where  $\hat{y}_3 = 1 - \hat{y}_1 - \hat{y}_2$ ) is given by Equation (22).

$$\mathcal{L}_{CE} = -\log(\hat{y}_1) \quad (22)$$

Graph: We display the cross-entropy loss for  $\hat{y}_1$  and  $\hat{y}_2$ , with  $\hat{y}_3$  determined implicitly. The hinge loss plot is presented in Figure 13.



**Figure 13.** Hinge loss.

Function: The hinge loss is graphed as a function of  $y \cdot \hat{y}$ . The hinge loss becomes zero when  $y \cdot \hat{y} \geq 1$  and grows linearly if it is less than that.

We visualize within the interval  $-2 \leq y \cdot \hat{y} \leq 2$ .

Hinge loss is commonly employed to train classifiers, such as support vector machines (SVMs), as shown in Equation (23).

$$\mathcal{L}_{\text{Hinge}} = \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i \hat{y}_i) \quad (23)$$

where:

- $n$  represents the total count of samples,
- $y_i$  denotes the actual label, which can be either  $-1$  or  $1$ ,
- $\hat{y}_i$  is the estimated label.

The Kullback–Leibler (KL) divergence quantifies the dissimilarity between a given probability distribution  $P$  and another reference probability distribution  $Q$  as shown in Equation (24).

$$\mathcal{L}_{\text{KL}} = \sum_{i=1}^n P(x_i) \log \frac{P(x_i)}{Q(x_i)} \quad (24)$$

where:

- $P(x_i)$  represents the actual probability distribution,
- $Q(x_i)$  is the estimated probability distribution.

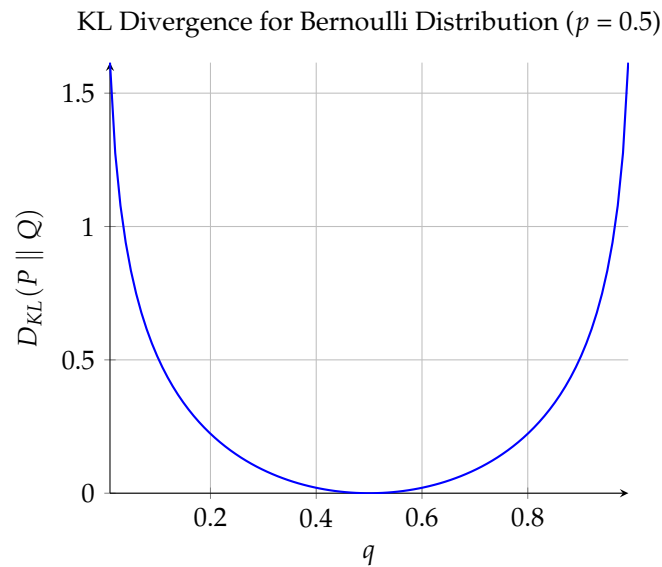
The Kullback–Leibler (KL) divergence plot is presented in Figure 14.

Definition: The Kullback–Leibler (KL) divergence for two Bernoulli distributions with success probabilities  $p$  and  $q$  is expressed as shown in Equation (25).

$$D_{\text{KL}}(P \parallel Q) = p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q} \quad (25)$$

Constant  $p$ : For this plot,  $p$  is set to a constant value of 0.5.

Variable  $q$ : The value of  $q$  is adjusted from 0.01 to 0.99 to circumvent the undefined points at 0 and 1.



**Figure 14.** Kullback–Leibler (KL) divergence.

Mean Absolute Error (MAE) loss calculates the mean of the absolute deviations between forecasted and true values as shown in Equation (26).

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (26)$$

where:

- $n$  represents the count of samples,
- $y_i$  represents the actual value,
- $\hat{y}_i$  is the value that is forecasted.

The MAE loss plot is presented in Figure 15.

Huber loss is more robust to outliers in data compared to Mean Squared Error (MSE). It merges the benefits of MSE and MAE by behaving quadratically for small errors and linearly for large errors, as shown in Equation (27).

$$\mathcal{L}_{\delta}(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \quad (27)$$

where:

- $y$  represents the actual value,
- $\hat{y}$  denotes the value that is forecasted,
- $\delta$  is the cutoff point where the loss changes from quadratic to linear.

The Huber loss plot is presented in Figure 16.

Below, some performance metrics for the lane-detection app using TensorFlow are presented.

Figure 17 shows the Confusion Matrix of the algorithm used for detecting lanes.

Figure 18 shows validation dataset F1 score.

Figure 19 shows the number of training epochs and their associated accuracy levels.

Figure 20 shows the precision of the lane-detection system.

Figure 21 shows the evaluation of the model using the Dice score.

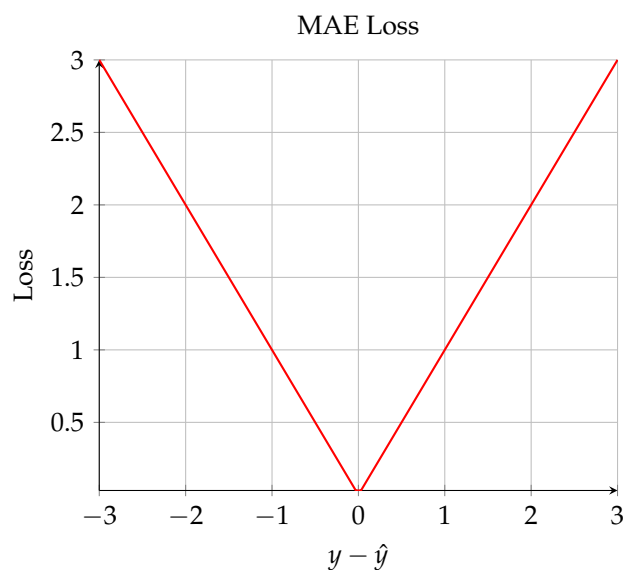


Figure 15. MAE loss.

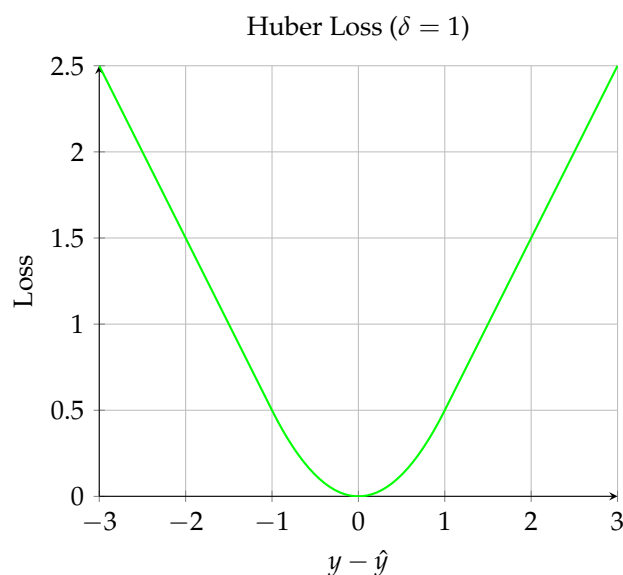


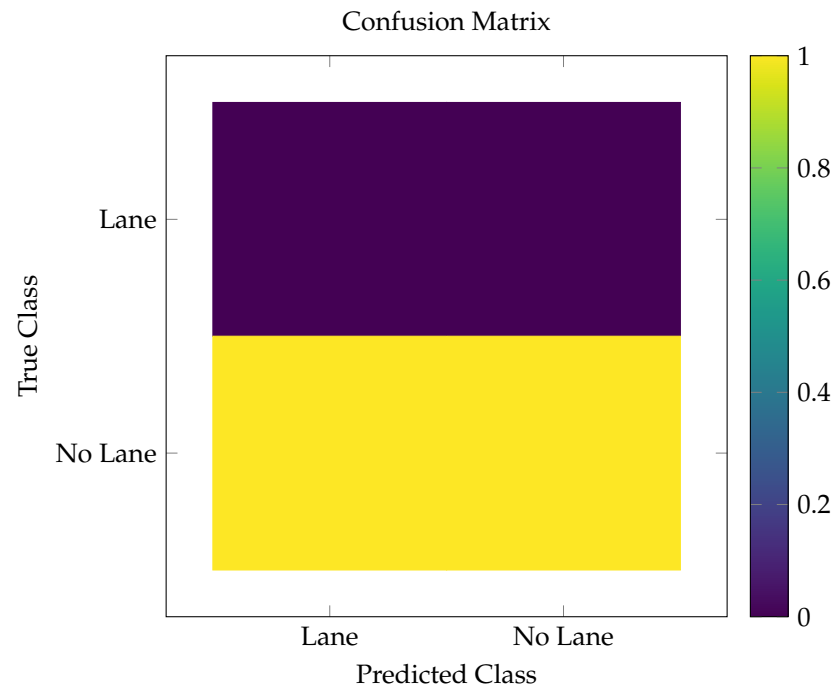
Figure 16. Huber loss.

Figure 22 shows a neural network simulation app, which is interesting for understanding how a neural network in TensorFlow is trained.

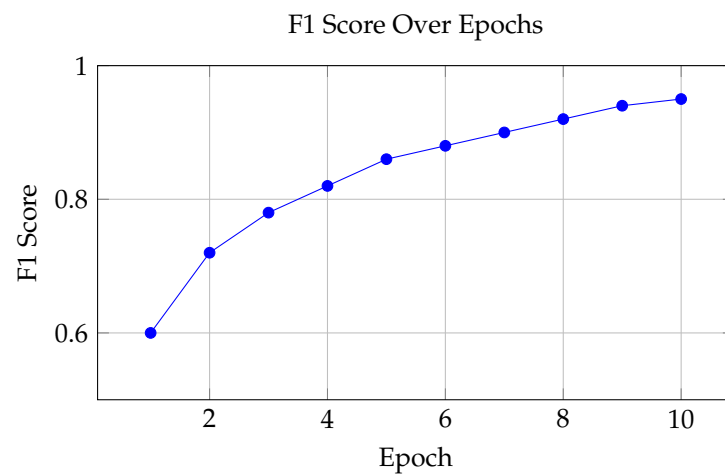
As seen in the neural network training app, the result was different every time the simulation was run, but the final result tended to be similar. Sometimes, it could not reach the desired result, but after a while, it would succeed. Sometimes, it was necessary to change some parameters in order to obtain the desired result, so the task when training the neural network is to configure the parameters in a way that the desired result will be reached the soonest as possible.

It was discovered that if the learning rate was set at a high value, then the system could reach the desired value many times faster, but there were many times when the system could not reach the desired value at all, so it is better to keep a medium learning rate, which could almost always reach the desired value, even if it takes a longer time.

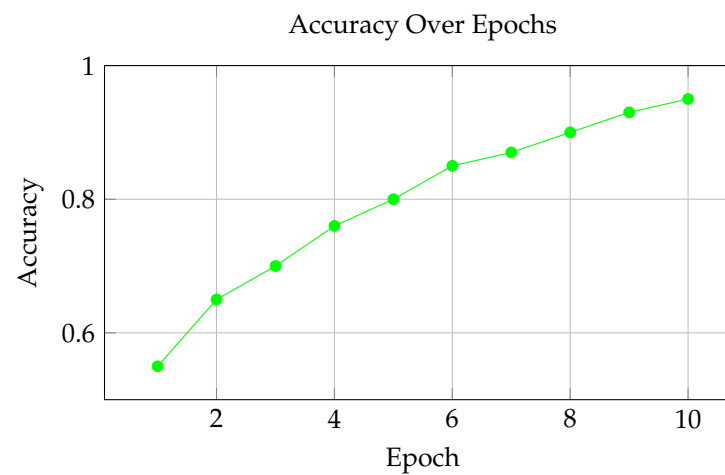
It was also discovered that the ratio of training to test data did not influence the results too much. In this situation, there is not too much difference. Even if the feature numbers or the number of neurons were increased, there was not too much influence, but the system worked slower.



**Figure 17.** Confusion Matrix of the algorithm used for detecting lanes.



**Figure 18.** Validation dataset F1 score.



**Figure 19.** Number of training epochs and their associated accuracy levels.

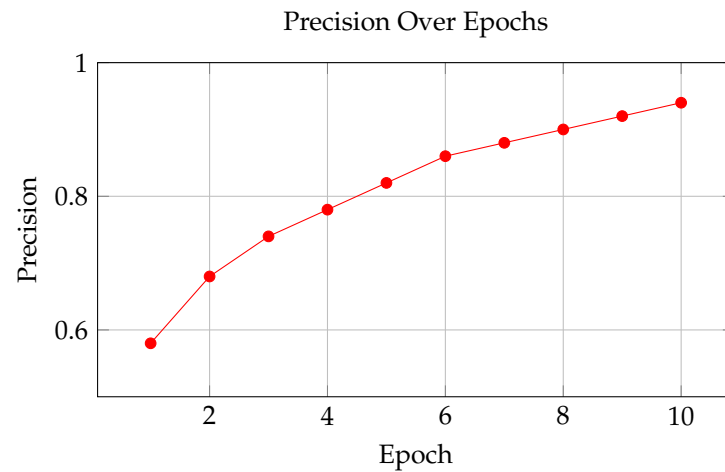


Figure 20. Precision of the lane-detection system.



Figure 21. Evaluation of the model using the Dice score.

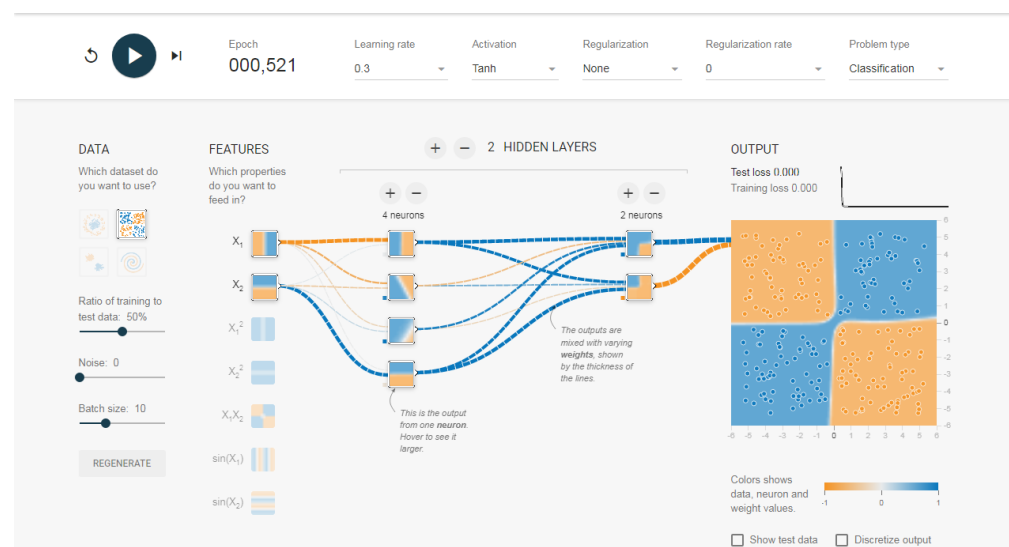


Figure 22. Trained neural network using TensorFlow [41].

It could also be seen that if the noise value was high, it was difficult for the system to obtain the desired values.

It was slow to show that if the batch size increased, the results were similar, but the time to obtain the desired results was longer.

It can be said that it is good to find the parameters to obtain the desired results almost at the same time.

Figure 23 shows that lane detection on curvy roads is carried out by the Android mobile-phone app.

Figure 24 shows the lane detection on a straight road made by the mobile-phone app on Android.

It can be seen that in both situations, the application works well, and the lane is detected. It can be seen that both mobile and side lanes on the road are detected.

These lines can guide a driver to stay on the road with the car, or it may even be possible to drive a self-driving car with this algorithm.

The software is configurable; the user can make changes, such as changing the color or the guidelines, which are overlaid in the live video feed. To do this, the user must click on the small gear button; this way, it will enable him to change the color of the guidelines, which detects the road lanes.

There can also be toggled whether one or both lanes are hidden and when only to have warnings for drivers to avoid being distracted by the app while driving.

The lane-detection app actually has two functionalities, which can be toggled using the gear button from the upper right corner. One functionality is the lane detection, and the other functionality is the car detection. The lane-detection app works better on empty country roads (Figures 23 and 24), while in congested traffic, the car-detection part of the app can be used, according to Figure 25.

The app can also be configured to detect cars on the road, also by tapping the gear icon (Figure 25).

The lane and car detection may seem opposite to each other, but both have some similarities. Both use image-recognition techniques using the OpenCV image-processing library on the Android operating system. Both use deep-learning algorithms, and the neural network is trained using TensorFlow.

The difference between the two parts of the app is that lane detection is used mainly in a low-traffic environment when we are going on an empty road, and we want to help keep the car on the road after a long drive when the driver is usually tired.

The car-detection part is used mainly in a high-traffic environment, when we want to detect other cars and obstacles, to avoid crashing into other cars, and also after a long drive when the driver is tired.

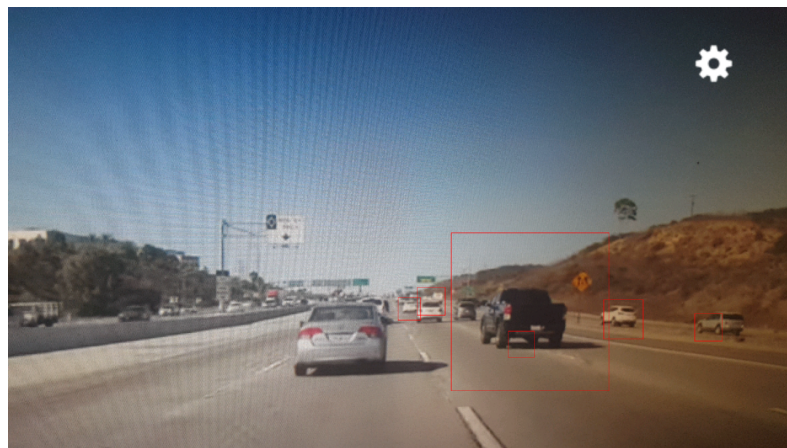
Both parts of the app can be used for a self-driving car app, where the self-driving car should avoid obstacles and keep the car on the road. If the app also includes traffic light and sign detection, it would be a complete self-driving car app.



**Figure 23.** Lane detection on a curved road made by the mobile-phone app.



**Figure 24.** Lane detection on a straight road made by the mobile-phone app.



**Figure 25.** Car detection made by the mobile-phone app.

#### 4.4. Results

The apps were posted on the Google Play Store, and some real user reviews and ratings will be presented next.

The apps are quite new, and many of them have no ratings or reviews on Google Play, like the tooth-brushing app, which has no ratings or reviews.

The night-vision app has a few reviews. Below are quoted a few of them:

- “Best app for night time”
- “Program error”
- “Doesn’t work”
- “Thank you for this AMAZING APP every other night-vision app was fake but thanks to you i can use it now, btw add colors”
- “Needs some light like how it gives a clear vision etc behind the tv”
- “ok close up but not good for distance”
- “Looked behind me TV really shows all. Needs some light but be sweet to check the engine etc.”

The average rating for the night-vision app is 3.

The lane-detection app has no reviews, but the average rating is 3.429.

These ratings are not the best, but if we search for more famous apps made by Google LLC, like the Blogger app, which has a rating of 3.7, and the Google Classroom app, which has a rating of 2.5, than the ratings of the presented apps it is not so bad. Many times, mobile app users are ruthless when they make reviews.



## 5. Discussions

### 5.1. Night-Vision App

This article presented a night-vision app for vehicles. The app can be easily used in any car to avoid collisions with animals or pedestrians crossing the road. The app is actually an image enhancer that can be used at night when there is poor visibility or bad weather.

Many expensive cars already have these features installed. This phone app is an alternative to having this premium feature in the car, with almost zero investment, based on the fact that almost everyone owns at least one smartphone.

If the user owns more smartphones, one old smartphone can be placed in the car to be used as a night-vision camera only.

The night-vision app is a valuable tool crafted for drivers who travel at night, seeking a more optimized smartphone experience within their vehicles. It eases the accessibility of key features and lessens distractions during nighttime journeys.

The app features a user interface tailored to lower eye strain in dim lighting, utilizing dark themes and minimizing glare. By prioritizing simplicity, it cuts down on unnecessary on-screen items, enabling drivers to concentrate on driving. It works with most modern smartphones and promotes safe navigation and smartphone handling by offering straight-forward options and decreasing the likelihood of distractions. It allows drivers to reach essential functions without juggling between multiple applications.

### 5.2. Tooth-Brushing Assistant App Using Image-Processing Techniques

As presented, a tooth-brushing app was created for the Android mobile-phone environment. The Android mobile-phone environment was chosen because it is the most widespread mobile-phone operating system.

The tooth-brushing app detected the correct tooth-brushing movements (up–down) using movement vectors, which were implemented using the OpenCV image-processing library.

The tooth-brushing app is a cutting-edge dental-hygiene application intended to enhance users' tooth-brushing behaviors. Suitable for all age groups, it assists users in adhering to correct methods and durations for superior oral care.

It features a detailed brushing guide accompanied by a timer to guarantee adequate coverage for all mouth regions. The app displays movement vectors to teach users effective brushing methods and dental-health approaches. It ensures proper tooth brushing, therefore decreasing the chances of cavities, gum issues, and plaque accumulation. Its appeal extends to both adults and children, promoting improved oral health habits for the entire family.

### 5.3. Lane- and Car-Detection App Using Deep-Learning Algorithms with TensorFlow

As presented, an app was created that could detect road lanes.

The uses OpenCV image-processing algorithm and TensorFlow are used to train the neural network for lane detection.

The app could configure the color of the overlaid line, which detects the road lane. The system can also be configured to hide one or both detected road lanes and to only issue warnings in the car that comes off the other side of the road. This way, the driver can focus on the road and not become distracted by the app.

The lane-detection application serves as a smart driving assistant aimed at enhancing road safety by assisting drivers in maintaining their lane position. It is particularly useful for extended journeys or when driving through unfamiliar areas, thus augmenting both ease and safety while driving.

Leveraging sophisticated algorithms, the app detects lane markings and offers real-time navigation assistance. It functions independently of continuous Internet connectivity, ensuring dependability even in remote locations. The risk of lane drifting, particularly due to fatigue or distractions, is significantly reduced. With dependable lane tracking, it boosts driver confidence when facing challenging conditions. Its user-friendly interface and uncomplicated functionality make it accessible to drivers of varying experience levels.

## 6. Conclusions

### 6.1. Night-Vision App

For the night-vision application, more image-enhancement functions were used from the OpenCV image-processing library. The innovation of this app is that it was made a night-vision app, even though the mobile phone does not have an infrared night-vision camera. The limitations are that it can obtain a better night-vision image with an external infrared night-vision camera than with a standard mobile-phone camera and image-processing techniques. The practical significance of the app is that it can be used quite well at night to see things in the dark.

The app was made only for the Android platform. Further enhancements could be to implement the app for the iOS platform, too. A good future study could be to implement the app for external thermal cameras, too. In this way, the warm bodies of humans and animals can be even better detected.

### 6.2. Tooth-Brushing Assistant App Using Image-Processing Techniques

For the tooth-brushing app, further enhancements can be made by adding a sensor to the toothbrush to detect the force the toothbrush applies to the teeth and gums. To protect the gums, it is not good to apply a large force to them during tooth brushing. When a strong force is applied to the gums, these can be harmed. The innovation of this app is that there are not many apps on the Google Play Store that actually can tell if a user brushes their teeth correctly just using image processing. Many apps connect to an external electric toothbrush via Bluetooth and check the direction of the tooth-brushing movement using an accelerometer. The innovation of this app is that it can work with any manual toothbrush. The limitations are that image processing can often present false negatives, usually when using toothpaste, which makes a lot of foam. The practical significance of the app is that it can be used quite well with any common toothbrush. There is no need to acquire an additional expensive electric toothbrush.

Another enhancement can be made to implement the app on the iOS operating system, too, to cover the other part of the mobile market.

### 6.3. Lane and Car-Detection App Using Deep-Learning Algorithms with TensorFlow

The lane-detection app can also be a starting point for autonomous cars. The innovation is that this app can detect lanes of cars with the usage of a mobile phone. There is no need to have this expensive feature that mostly premium cars have. The limitations are that the system uses only one camera from the mobile phone, so if it is accidentally covered, then the system cannot detect lanes or cars. The system is external, so it has no feedback to make the steering wheel vibrate when lanes are not followed or apply the brakes of the car to avoid an accident when using the car-detection algorithm. The practical significance of the app is that a cheap mobile phone can be used to achieve a premium feature.

Future enhancements can also be implemented to implement the app on the iOS operating system, i.e., to cover the rest of the mobile-phone users.

Another enhancement can also be adding traffic light and sign detection to build a complete self-driving car application.

**Funding:** This research was funded by MDPI (Multidisciplinary Digital Publishing Institute).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All the images are available on request from the corresponding author.

**Acknowledgments:** The author would like to thank Politehnica University Timisoara, Romania for the given support.

**Conflicts of Interest:** The author declares no conflicts of interest.

## Abbreviations

The following symbols and abbreviations are used in this manuscript:

GPS	Global Positioning System
HUD	Head-Up Display
SIM	Subscriber Identity Module
4G, 5G	fourth, fifth generation of broadband cellular network technology
eSIM	embedded-SIM (Subscriber Identity Module)
OpenCV	Open Computer-Vision Library
ADAS	advanced driver-assistance systems
CNNs	Convolutional Neural Networks
LiDAR	Light Detection And Ranging or Laser imaging, Detection, And Ranging
GPUs	Graphics Processing Units
iOS	iPhone Operating System
IR	Infra Red
IoT	Internet of Things
AR	Augmented Reality
$x, y$	spatial coordinates
$I(x, y)$	input image
NV	Night Vision
$k$	amplification factor constant
$T$	threshold value
$\vec{v}$	velocity vector
$\frac{dx}{dt}, \frac{dy}{dt}$	horizontal and vertical components of velocity
$\vec{F}$	optical flow
$u, v$	horizontal and vertical components of optical flow
$\vec{M}$	movement of objects
$dx, dy$	spatial displacement
$dt$	temporal displacement
$\vec{G}$	global motion
$tx, ty$	translational motion in the horizontal and vertical directions
$A_{cc}$	accuracy
$C_{clip}$	number of estimated lane points that are correctly managed
$S_{clip}$	number of truth ground points
$F_1$	Precision and Recall function
TP	Total Points
FP	False Positives
FN	False Negatives
$Z^{[l]}$	convolutional layer's output
Conv_2D	convolutional filter
$A^{[l-1]}$	activation of the preceding layer
$A^{[l]}$	activation of layer $l$
$l$	convolutional layer
ReLU	Rectified Linear Unit

$W^{[l]}$	weights
$b^{[l]}$	bias vector
$\hat{y}$	network's last layer
MSE	Mean Squared Error
$n$	total count of samples
$y_i$	actual value, binary true label, either $-1$ or $1$
$\hat{y}_i$	predicted value, the estimated likelihood of belonging to the positive class
$k$	count of classes
$y_{ij}$	binary indicator (0 or 1) indicating whether the class label $j$ correctly classifies sample $i$
$\hat{y}_{ij}$	forecasted likelihood that sample $i$ belongs to class $j$
SVMs	support vector machines
KL	Kullback–Leibler
$P(x_i)$	actual probability distribution
$Q(x_i)$	estimated probability distribution
MAE	Mean Absolute Error
$\delta$	cutoff point where the loss changes from quadratic to linear

## References

- Nowosielski, A.; Małecki, K.; Forczmański, P.; Smoliński, A.; Krzywicki, K. Embedded Night-Vision System for Pedestrian Detection. *IEEE Sens. J.* **2020**, *20*, 9293–9304. [[CrossRef](#)]
- Purohit, M.; Chakraborty, A.; Kumar, A.; Kaushik, B.K. Image Processing Framework for Performance Enhancement of Low-Light Image Sensors. *IEEE Sens. J.* **2021**, *21*, 8530–8542. [[CrossRef](#)]
- Spivak, A.; Belenky, A.; Fish, A.; Yadid-Pecht, O. A Wide-Dynamic-Range CMOS Image Sensor with Gating for Night Vision Systems. *IEEE Trans. Circuits Syst. II Express Briefs* **2011**, *58*, 85–89. [[CrossRef](#)]
- Liu, Z.; Blasch, E.; Xue, Z.; Zhao, J.; Laganieri, R.; Wu, W. Objective Assessment of Multiresolution Image Fusion Algorithms for Context Enhancement in Night Vision: A Comparative Study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 94–109. [[CrossRef](#)] [[PubMed](#)]
- Bhatnagar, G.; Wu, Q.M.J. A fractal dimension based framework for night vision fusion. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 220–227. [[CrossRef](#)]
- Wang, Y.; Zhang, Y.; Guo, Q.; Zhao, M.; Jiang, Y. RNVE: A Real Nighttime Vision Enhancement Benchmark and Dual-Stream Fusion Network. *IEEE Signal Process. Lett.* **2024**, *31*, 131–135. [[CrossRef](#)]
- Kim, J.; Jeon, M.-H.; Cho, Y.; Kim, A. Dark Synthetic Vision: Lightweight Active Vision to Navigate in the Dark. *IEEE Robot. Autom. Lett.* **2021**, *6*, 143–150. [[CrossRef](#)]
- Warrant, E.; Oskarsson, M.; Malm, H. The Remarkable Visual Abilities of Nocturnal Insects: Neural Principles and Bioinspired Night-Vision Algorithms. *Proc. IEEE* **2014**, *102*, 1411–1426. [[CrossRef](#)]
- Bandani, A.K.; Bollampally, A.; Sahithi, S.; Naik, R.; Kumar, N.; Goutham. Design of Spy Robot with Wireless Night Vision Camera Using Android. In Proceedings of the 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 24–26 January 2023; pp. 1–5.
- Valentino, A.M.; Leonen, J.T. IoT-Based Smart Security Robot with Android App, Night Vision and Enhanced Threat Detection. In Proceedings of the 2023 IEEE 15th International Conference on Computational Intelligence and Communication Networks (CICN), Bangkok, Thailand, 22–23 December 2023; pp. 415–420.
- Herath, B.; Dewmin, G.H.S.; Sukumaran, S.; Amarasinghe, Y.W.R.; De Silva, A.H.T.E.; Mitani, A.; Wijethunge, D.; Sampath, W.H.P. Design and Development of a Novel Oral Care Simulator for the Training of Nurses. *IEEE Trans. Biomed. Eng.* **2020**, *67*, 1314–1320. [[CrossRef](#)]
- Lee, Y.-J.; Lee, P.-J.; Kim, K.-S.; Park, W.; Kim, K.-D.; Hwang, D.; Lee, J.-W. Toothbrushing Region Detection Using Three-Axis Accelerometer and Magnetic Sensor. *IEEE Trans. Biomed. Eng.* **2012**, *59*, 872–881.
- Wang, M.-Y.; Yang, T.-H.; Huang, H.; Hsu, H.-Y.; Kuo, L.-C.; Su, F.-C.; Huang, C.-C. Evaluation of Hand Tendon Movement by Using High-Frequency Ultrasound Vector Doppler Imaging. *IEEE Trans. Biomed. Eng.* **2020**, *67*, 2945–2952. [[CrossRef](#)] [[PubMed](#)]
- Xing, F.; Woo, J.; Gomez, A.D.; Pham, D.L.; Bayly, P.V.; Stone, M.; Prince, J.L. Phase Vector Incompressible Registration Algorithm for Motion Estimation From Tagged Magnetic Resonance Images. *IEEE Trans. Med. Imaging* **2017**, *36*, 2116–2128. [[CrossRef](#)] [[PubMed](#)]
- Kruizinga, P.; Mastik, F.; Bosch, J.G.; de Jong, N.; van der Steen, A.F.W.; Van Soest, G. Measuring submicrometer displacement vectors using high-frame-rate ultrasound imaging. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control.* **2015**, *62*, 1733–1744. [[CrossRef](#)] [[PubMed](#)]

16. Ouyang, Z.; Hu, J.; Niu, J.; Qi, Z. An Asymmetrical Acoustic Field Detection System for Daily Tooth Brushing Monitoring. In Proceedings of the IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
17. Lutze, R. Practicality of Smartwatch Apps for Supporting Elderly People—A Comprehensive Survey. In Proceedings of the IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Stuttgart, Germany, 17–20 June 2018; pp. 1–7.
18. Demrozi, F.; Jereghi, M.; Pravadelli, G. Towards the automatic data annotation for human activity recognition based on wearables and BLE beacons. In Proceedings of the IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), Kailua-Kona, HI, USA, 22–25 March 2021; pp. 1–4.
19. Jung, S.; Youn, J.; Sull, S. Efficient Lane Detection Based on Spatiotemporal Images. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 289–295. [[CrossRef](#)]
20. Wang, Y.; Jing, Z.; Ji, Z.; Wang, L.; Zhou, G.; Gao, Q.; Zhao, W.; Dai, S. Lane Detection Based on Two-Stage Noise Features Filtering and Clustering. *IEEE Sens. J.* **2022**, *22*, 15526–15536. [[CrossRef](#)]
21. Chen, S.; Huang, L.; Chen, H.; Bai, J. Multi-Lane Detection and Tracking Using Temporal-Spatial Model and Particle Filtering. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2227–2245. [[CrossRef](#)]
22. Maghsoumi, H.; Masoumi, N.; Araabi, B.N. RoadSaVe: A Robust Lane Detection Method Based on Validity Borrowing From Reliable Lines. *IEEE Sens. J.* **2023**, *23*, 14571–14582. [[CrossRef](#)]
23. Lu, P.; Cui, C.; Xu, S.; Peng, H.; Wang, F. SUPER: A Novel Lane Detection System. *IEEE Trans. Intell. Veh.* **2021**, *6*, 583–593. [[CrossRef](#)]
24. Mukhopadhyay, A.; Murthy, L.R.D.; Mukherjee, I.; Biswas, P. A Hybrid Lane Detection Model for Wild Road Conditions. *IEEE Trans. Artif. Intell.* **2023**, *4*, 1592–1601. [[CrossRef](#)]
25. Feng, Y.; Li, J. Robust Accurate Lane Detection and Tracking for Automated Rubber-Tired Gantries in a Container Terminal. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 11254–11264. [[CrossRef](#)]
26. Maghsoumi, H.; Masoumi, N.; Araabi, B.N. Lane Detection and Tracking Datasets: Efficient Investigation and New Measurement by a Novel “Dataset Scenario Detector” Application. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 5007716. [[CrossRef](#)]
27. Luo, S.; Zhang, X.; Hu, J.; Xu, J. Multiple Lane Detection via Combining Complementary Structural Constraints. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 7597–7606. [[CrossRef](#)]
28. Zhang, J.; Deng, T.; Yan, F.; Liu, W. Lane Detection Model Based on Spatio-Temporal Network With Double Convolutional Gated Recurrent Units. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6666–6678. [[CrossRef](#)]
29. Wang, H.; Liu, B. G-NET: Accurate Lane Detection Model for Autonomous Vehicle. *IEEE Syst. J.* **2023**, *17*, 2039–2048. [[CrossRef](#)]
30. Lee, C.; Moon, J.-H. Robust Lane Detection and Tracking for Real-Time Applications. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 4043–4048. [[CrossRef](#)]
31. Dewangan, D.K.; Sahu, S.P. Driving Behavior Analysis of Intelligent Vehicle System for Lane Detection Using Vision-Sensor. *IEEE Sens. J.* **2021**, *21*, 6367–6375. [[CrossRef](#)]
32. Wang, Q.; Han, T.; Qin, Z.; Gao, J.; Li, X. Multitask Attention Network for Lane Detection and Fitting. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 1066–1078. [[CrossRef](#)]
33. Wu, C.-B.; Wang, L.-H.; Wang, K.-C. Ultra-Low Complexity Block-Based Lane Detection and Departure Warning System. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 582–593. [[CrossRef](#)]
34. Sivaraman, S.; Trivedi, M.M. Integrated Lane and Vehicle Detection, Localization, and Tracking: A Synergistic Approach. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 906–917. [[CrossRef](#)]
35. Zhang, W. Combination of SIFT, Canny Edge Detection for Registration Between SAR and Optical Images. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 4007205. [[CrossRef](#)]
36. Ding, Y.; Sun, Y.; Yu, X.; Cheng, D.; Lin, X.; Xu, X. Bezier-Based Hough Transforms for Doppler Localization of Human Targets. *IEEE Antennas Wirel. Propag. Lett.* **2020**, *19*, 173–177. [[CrossRef](#)]
37. Yang, W.; Luo, H.; Tse, K.-W.; Hu, H.; Liu, K.; Li, B.; Wen, C.-Y. Autonomous—Targetless Extrinsic Calibration of Thermal, RGB, and LiDAR Sensors. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 4510011. [[CrossRef](#)]
38. Talpes, E.; Sarma, D.D.; Venkataramanan, G.; Bannon, P.; McGee, B.; Floering, B.; Jalote, A.; Hsiong, C.; Arora, S.; Gorti, A.; et al. Compute Solution for Tesla’s Full Self-Driving Computer. *IEEE Micro* **2020**, *40*, 25–35. [[CrossRef](#)]
39. Ettinger, S.; Cheng, S.; Caine, B.; Liu, C.; Zhao, H.; Pradhan, S.; Chai, Y.; Sapp, B.; Qi, C.; Zhou, Y.; et al. Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 9690–9699.

40. Shyam, P.; Yoon, K.-J.; Kim, K.-S. Weakly Supervised Approach for Joint Object and Lane Marking Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, BC, Canada, 11–17 October 2021; pp. 2885–2895.
41. Tensor Flow Playground. Available online: <https://playground.tensorflow.org> (accessed on 3 July 2024).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.