applied sciences

MDPI

Article

AI on the Road: NVIDIA Jetson Nano-Powered Computer Vision-Based System for Real-Time Pedestrian and Priority Sign Detection

Kornel Sarvajcz [1], Laszlo Ari [1] and Jozsef Menyhart [2,*]

[1] Mechatronics Department, Vehicles and Mechatronics Institute, Faculty of Engineering, University of Debrecen, Ótemető Str. 2-4, 4028 Debrecen, Hungary; sarvajcz@eng.unideb.hu (K.S.)
[2] Department of Vehicles Engineering, Vehicles and Mechatronics Institute, Faculty of Engineering, University of Debrecen, Ótemető Str. 2-4, 4028 Debrecen, Hungary
* Correspondence: jozsef.menyhart@eng.unideb.hu

Abstract: Advances in information and signal processing, driven by artificial intelligence techniques and recent breakthroughs in deep learning, have significantly impacted autonomous driving by enhancing safety and reducing the dependence on human intervention. Generally, prevailing ADASs (advanced driver assistance systems) incorporate costly components, making them financially unattainable for a substantial portion of the population. This paper proposes a solution: an embedded system designed for real-time pedestrian and priority sign detection, offering affordability and universal applicability across various vehicles. The suggested system, which comprises two cameras, an NVIDIA Jetson Nano B01 low-power edge device and an LCD (liquid crystal system) display, ensures seamless integration into a vehicle without occupying substantial space and provides a cost-effective alternative. The primary focus of this research is addressing accidents caused by the failure to yield priority to other drivers or pedestrians. Our study stands out from existing research by concurrently addressing traffic sign recognition and pedestrian detection, concentrating on identifying five crucial objects: pedestrians, pedestrian crossings (signs and road paintings separately), stop signs, and give way signs. Object detection was executed using a lightweight, custom-trained CNN (convolutional neural network) known as SSD (Single Shot Detector)-MobileNet, implemented on the Jetson Nano. To tailor the model for this specific application, the pre-trained neural network underwent training on our custom dataset consisting of images captured on the road under diverse lighting and traffic conditions. The outcomes of the proposed system offer promising results, positioning it as a viable candidate for real-time implementation; its contributions are noteworthy in advancing the safety and accessibility of autonomous driving technologies.

Keywords: ADAS; AI; CNN; computer vision; deep learning; embedded system; neural networks; PyTorch; real-time object detection; transfer learning

Citation: Sarvajcz, K.; Ari, L.; Menyhart, J. AI on the Road: NVIDIA Jetson Nano-Powered Computer Vision-Based System for Real-Time Pedestrian and Priority Sign Detection. *Appl. Sci.* **2024**, *14*, 1440. https://doi.org/10.3390/app14041440

Academic Editor: Suchao Xie

Received: 20 December 2023
Revised: 1 February 2024
Accepted: 2 February 2024
Published: 9 February 2024

1. Introduction

Population growth contributes to a surge in the number of vehicles on the road, consequently elevating the risk of accidents stemming from factors such as fatigue, drowsiness, drunkenness, distraction, and road conditions [1]. The daily flow of traffic constitutes a multifaceted process demanding shared attention, swift reaction time, and sound judgment. A significant portion of accidents can be attributed to prevalent causes, including insufficient speed selection and a failure to yield priority [2].

The pursuit of developing systems to enhance the daily lives of individuals has been a longstanding research objective across various fields. Among the most impactful innovations for humanity are vehicles. To attain complete mastery over their vehicles, individuals require artificial systems to assist them. Numerous studies and solutions have emerged to enhance the road safety of both drivers and pedestrians. These technological advancements

form the basis of advanced driver assistance systems (ADASs) and autonomous driving systems. ADASs are designed to aid drivers and vehicles in identifying potentially hazardous traffic situations and responding promptly and accurately [3]. The growth of safety and comfort-enhancing technologies in the automotive sector has led to a pronounced focus on the study and development of these systems. They have evolved into a pivotal area of research due to their capacity to enhance the overall driving experience. While various technologies contribute to the creation of these systems [4], camera-based solutions offer significant cost advantages along with the ability to process and analyze images using the latest computer vision technologies [5]. External environment sensing involves the collection of information about the driving environment [6], compassing nearby vehicles, pedestrians, traffic signs, traffic lights, and various objects [7]. The design of such systems must consider various factors, including weather, road, and lighting conditions, as these elements can significantly impact the perceptual quality of environmental information [8].

Over the past decade, there has been a noteworthy emphasis on the development of traffic surveillance systems, directed towards enhancing safety through the real-time monitoring of the road environment. The availability of multiple sensing modalities, such as radar, LIDAR (light detection and ranging), and cameras, coupled with the increase in computing power, has empowered engineers to devise increasingly sophisticated solutions for real-time computer vision-based assistance systems [9]. Significant strides in vehicular sensor technology have been documented, enabling the execution of various simple and complex tasks, including object detection [10], localization [11], tracking [12], and activity recognition [13] across a diverse array of applications.

2. CNN Deployment in Automotive

As technology advances, the expectations of car buyers evolve, with a growing demand for vehicles that seamlessly integrate with their digital lives while prioritizing safety. Automakers are undergoing a paradigm shift from emphasizing horsepower to prioritizing digital technology, particularly in the creation of self-driving cars, commonly referred to as autonomous driving systems (ADSs). Anticipated in the upcoming years is the next generation of ADSs, poised to deliver an elevated level of self-driving experience. To achieve this, various technologies must be seamlessly integrated into a singular system capable of communication, collaboration, and ultimately emulating human functions across almost all driving scenarios. Intel suggests [14] that achieving this level of sophistication necessitates the incorporation of several sensors such as radar, LIDARs, GPS (Global Positioning System), and advanced cameras. These sensors are tasked with collecting crucial information at a rate of approximately 1 GB/second and processing it in real-time to ensure sound driving decisions. While LIDARs offer advantages like high precision and 3D mapping, their high cost renders them impractical for widespread commercial use in the industry. Consequently, some companies, including Tesla [15] and Mobileye [16], have opted for optical cameras and radars as their preferred sensing devices in the vision-based components of ADSs. In this context, convolutional neural networks (CNNs), as an image-based technique, have demonstrated remarkable effectiveness and are widely employed by automakers in the object detector modules of vehicles.

Despite the notable advancements in object detection using convolutional neural networks (CNNs) for various applications, the implementation of such networks in autonomous driving systems (ADSs) remains a considerable challenge. This challenge arises from the stringent real-time constraints imposed on safe decision-making and data processing latency within ADSs. As a result, the development of an end-to-end ADS design has not been fully resolved, and ADSs continue to be subjects of ongoing experimentation [17].

2.1. CNN-Based Object Detection

CNNs belong to the family of deep neural networks (DNNs), characterized by artificial neurons organized in layers. They possess distinctive attributes setting them apart from other types of neural networks, including local receptive fields, spatial subsampling, and

shared weights. CNNs facilitate cooperation between multiple sequential stages, involving convolution, padding, pooling operations, and the introduction of nonlinearities via activation functions (Sigmoid, Softmax, TanH, ReLU, Leaky ReLU) and fully connected layers (FCLs). Each convolutional layer has an input and output called feature maps, which are 2D arrays of color channels for images and 3D arrays for videos, and 1D arrays for audio inputs. Throughout the network's processing of input data, new features are extracted at each output stage, and convolutional operations detect more distinctive features in subsequent layers. T. Turay et al. [18] offer a comprehensive definition of all architectural components of a CNN, including their mathematical equations, and employ the reference architecture of AlexNet [19].

Object detection plays a pivotal role in enabling self-driving cars and ADASs to perceive their environment and make well-informed decisions. Various detection algorithms, including deep CNN approaches, have been proposed for this purpose. However, these models often pose significant computational demands, rendering them unsuitable for deployment on embedded platforms with limited resources. Hence, there exists an urgent necessity to develop a more compact neural network model capable of addressing this computational challenge. The primary challenge in this context is to ensure the successful transplantation of the object detection network model onto an embedded platform without compromising accuracy and stability, which is crucial for the safe and reliable operation of self-driving cars and ADASs.

Object detection algorithms are prominently categorized into either one-stage or two-stage algorithms. Only one-stage algorithms possess the capability to operate efficiently under the constraints of embedded systems in real-time. In contrast, two-stage algorithms, exemplified by Faster R-CNN [20], demand more computational resources, rendering them unsuitable for deployment on embedded platforms. Presently, SSD (Single Shot Detector) [21] and YOLO (You Only Look Once) [22] stand out as two widely utilized one-stage object detectors. According to W. Liu et al. [21], YOLO may not be as effective in detecting dense multiple targets and large objects, while the SSD detector is deemed more suitable for effectively detecting a large number of objects with varying scales and types.

2.2. The Single Shot Detector (SSD)

The Single Shot Detector employs a feed-forward convolutional network to generate a pre-established set of bounding boxes and scores for object detection. The base network is designed for high-quality image classification and forms the initial layers of this network. It is followed by one or more convolutional layers on existing feature maps to predict a predefined number of bounding boxes along with corresponding confidence scores. A non-maximum suppression step is then applied to these detections to obtain the final results. The predicted boxes are displaced concerning predetermined and fixed prior boxes, positioned across a feature map and at the midpoint of each cell [21].

When training a detector using the SSD [21] method, the key distinction from training a typical detector employing region proposals lies in the necessity to assign ground-truth information to specific outputs within the fixed set of detector outputs. Following this assignment, the loss function and backpropagation are applied end-to-end. The training process involves selecting the set of default boxes and scales for detection, as well as the hard negative mining and data augmentation strategies. Throughout training, the network is tasked with determining which default boxes correspond to a ground-truth detection and adjusting accordingly. Each ground-truth box is matched to the default box with the highest Jaccard overlap, and default boxes are matched to any ground truth with Jaccard overlap exceeding a threshold of 0.5. This approach simplifies the learning process, enabling the network to predict high scores for multiple overlapping default boxes, as opposed to mandating the selection of only the one with maximum overlap, as seen in the MultiBox method.

During the training process, the SSD [21] focuses on optimizing two types of loss: localization (Lloc) and confidence loss (Lconf), which are then combined into a weighted

sum (where α represents the weight term) to obtain the overall loss function (L) for N matched prior boxes. The localization loss is dedicated to regressing the offsets between the prior boxes and the model's predictions, considering both the position (center) and size (width, height) of the object. Simultaneously, the confidence loss is calculated using a Softmax with cross-entropy loss. For more in-depth details about these two loss functions, further information can be found in reference [21].

$$L = \frac{1}{N} * (L_{conf} + \alpha * L_{loc}), \quad (1)$$

2.3. Backbone Networks

Classification algorithms are utilized as a backbone or base network in object detection models, functioning as fundamental feature extractor. In the following sub-sections we will delve into the most popular lightweight backbone models explicitly designed for object detection.

2.3.1. VGG16 Backbone Network

The VGG16-SSD (visual geometry group with 16 layers) network architecture employs the VGG16 network [23] as its backbone network model. It generates six feature maps with different dimensions, enabling object detection across various scales. Subsequently, a non-maximum suppression process is applied on the detection outputs, where the output with the highest confidence score is chosen as the definitive detection result for each group of overlapping detections. However, the VGG16 network's substantial architecture renders it impractical for embedded platforms, where it may exceed the system memory limit and cause difficulty achieving real-time performance.

2.3.2. MobileNet Backbone Network

Howard et al. [24] from Google have replaced the VGG16 backbone network with the MobileNet network model to enhance the real-time performance of the SSD detector and reduce its computational complexity. The second-generation Mobilenet-v2 is used as the backbone network model of the SSD detector in the existing MobileNet-SSD network architecture, inheriting the design principles from VGG16-SSD. The Mobilenet-v2 front-end network produces six feature maps with varying dimensions, enabling the back-end detection network to perform multi-scale object detection. The utilization of depth-wise separable convolutions effectively reduces the number of parameters compared to a convolution structure with the same spatial dimensions. This alteration empowers the MobileNet-SSD detector to achieve real-time performance, making it faster than other existing object detection networks [25].

2.3.3. MobileNetV2 Backbone Network

The backbone network for most lightweight network models is Mobilenet-v2, characterized by a standard convolutional layer and 17 inverse residual modules. Each module contains a 1×1 convolutional layer, a 3×3 depth-wise separable convolutional layer, batch normalization, and Relu6 (Rectified Linear Unit) excitation functions. The incorporation of inverse residual modules helps to prevent gradient-vanishing issues and allows for effective training during the backpropagation process. In traditional MobileNet-SSD, downsampling the feature map by a 1/16-scale convolution layer can result in poor and unstable detection of small objects in images. To overcome this limitation, Chiu et al. [25] devised a solution by extracting feature maps at 1/16 and 1/32 scales and adding four inverse residual modules to extract feature maps at scales of 1/64, 1/128, 1/256, and 1/512. The inclusion of the FPN (Feature Pyramid Network) module further enhances the feature maps with different scales.

2.4. Libraries and Frameworks

Overview of the CNN Libraries and Frameworks, typically Python-based for flexibility and productivity, cover data pre-processing and CNN algorithm construction. Several libraries such as NumPy [26], SciPy [27], and Pandas [28], Matplotlib [29] are used for data pre-processing. For the second stage, there are many frequently used libraries and frameworks. The most popular ones are the following.

2.4.1. Tensorflow

TensorFlow [30] is a popular open-source machine learning library introduced by Google in 2015. It uses data flow graphs where edges represent tensors to facilitate the development of computationally intensive machine learning tasks. Its API (application programming interface) is adept at distributing workloads across diverse nodes, including GPUs (graphics processing units) and CPUs (central processing units).

2.4.2. Pytorch

PyTorch [31], developed by Facebook AI Research Lab, stands out as a rapidly advancing library distinguished by innovative features such as distributed training scalability and cloud support, integrating seamlessly with NumPy. It offers design simplicity across the entire research-to-production spectrum; since March 2019, it has garnered recognition as the preeminent machine learning framework according to open literature [32].

2.4.3. Caffe

Caffe [33] is a research-focused library developed by the Berkeley AI Research (BAIR) team and community contributors. It is written in C++ and has a Python interface (2.7 or 3.3+). Caffe is distinguished by its remarkable processing speed and facilitates effortless transitions between CPU and GPU operations through a single flag. Furthermore, it seamlessly integrates with GPU training for image-based datasets.

2.4.4. Keras

Keras [34] is a high-level API (application programming interface) designed to operate atop Theano or Tensorflow, making it a user-friendly and widely adopted library. It offers a productive and easy-to-use interface that reduces the cognitive burden of developers.

3. Related Works

There are increasing studies in the development of intelligent monitoring systems, such as the fatigue and alertness monitoring system. In [35], a specialized system was devised to monitor the physiological status of drivers, employing computer vision and neural networks. The developed system continuously checks the driver's activities through cabin-installed cameras and promptly issuing warning signals and alarms upon detecting any abnormal driving behaviors or actions. The cameras specifically monitor the driver's head and eye pose, as well as eye rotation. The acquired information and signals undergo analysis via an echo state network (ESN). This neural network runs on an NVIDIA Jetson Nano. While the system attains an accuracy of 98%, perfection remains elusive.

The perpetual challenge confronting computer vision-based systems lies in the dynamic nature of visual conditions. In [36], a code was devised to assess both the condition of the road surface and the prevailing weather conditions, like dry, wet, and snowy. This system relies on the vehicle's camera system, operating with a custom-trained neural network (VGG16).

Recent research has also focused on studying road surface markings integral to traffic signalization. In their work, Ye et al. [37] proposed a two-stage model employing YOLO-v2 [22] to address the challenge posed by partially distorted and worn road markings. In the initial stage, the YOLO-v2 CNN was tasked with detecting the primary road markings, and in the second stage, a lightweight, and transformation-invariant classification network, RM-Net, which can handle distortions and surface wear, was utilized to identify road

markings. The authors have developed an annotated road marking detection dataset consisting of approximately 12,000 high-resolution images categorized into 13 classes, captured under diverse weather conditions during both day and night. The achieved result of the model is notably commendable with an 86.5% mAP on the dataset, thereby surpassing the performance of other existing frameworks.

L. Barba-Guaman et al. [38] developed an application dedicated to the recognition of pedestrians and vehicles. This system operates in conjunction with an NVIDIA Jetson Nano edge device. Their study incorporates five distinct pre-trained models, namely PedNet, MultiPed, SSD-MobileNet v1 and v2, and SSD-Inception v2.

The detection of traffic signalization, encompassing vital road features like traffic signs, traffic lights, and road surface markings, within the vicinity of self-driving cars stands as a fundamental prerequisite for the reliable operation of autonomous driving systems. This crucial feature ensures compliance with traffic regulations and elevates both vehicle and occupant safety. Consequently, an array of recent studies has focused on this research domain, aiming to further improve the robustness and reliability of traffic signal detection algorithms and systems. Henchri and Mtibaa [39] proposed a two-stage approach for traffic sign detection. In the initial stage, detection and classification are limited to the circular or triangular shapes of the signs, achieved through HOG (Histogram of Oriented Gradients) [40] features and support vector machines (SVM). Subsequently, in the second stage, a CNN is employed to classify these detected shapes into their respective subclasses. Finally, the approach is evaluated on the GTSDB (German Traffic Sign Detection Benchmark) [41], showing improved results.

In their system, J. Müller et al. [42] employed an SSD [21] detector and adapted the Inception-v3 CNN as the base network, instead of the originally enlisted VGG CNN, to enhance both speed and accuracy. The study introduced modifications to the prior box generation, allowing smaller strides in the latter network layers, thereby improving the detection of smaller objects. Non-maximum suppression (NMS) was also incorporated to avoid multiple detections of a single object. Moreover, an additional block was integrated to classify the states of the traffic lights (red, amber, or green). The model demonstrated excellent performance when evaluated on the DriveU [43] traffic light dataset.

E. Güney et al. [44] introduced an advanced driver assistance system (ADAS) designed for real-time detection of traffic signs, vehicles, and pedestrians. The system is portable and image-based, and uses the YOLO v5 algorithm to achieve high detection speed and accuracy. The training process involved a fusion of the German Traffic Sign Recognition Benchmark (GTSRB) dataset and a dataset tailored for the study, intentionally designed to assess system performance under diverse lighting and weather conditions. The model underwent training on a Tesla P100 graphics processing unit (GPU) using nearly 2500 images over a period of 8 h. To evaluate the performance of the model, various implementation metrics such as F1 score, precision (P), recall (R), and precision-recall (PR) curves were computed. Furthermore, the real-time performance of the model was systematically compared across different low-power, high-performance embedded platforms and on a test computer. The results indicated that the Jetson Xavier AGX platform achieved the highest real-time detection speed, recording 43.59 frames per second.

R. Ayachi et al. [45] introduced a traffic sign recognition application that achieves a good trade-off between speed and accuracy, rendering it well-suited for embedded implementations. The proposed approach for traffic sign recognition rested on a tailored CNN featuring three convolution layers, a non-linear activation layer, batch normalization layer, and max-pooling layer to compress feature maps. The CNN was scaled to balance different dimension parameters, resulting in a compact model size suitable for embedded implementation. The technique proved efficient, as the proposed CNN achieved an impressive accuracy of 99.3% upon evaluation using the European traffic sign recognition dataset; the real-time processing capability was demonstrated at 250 frames per second (FPS) when implemented on an NVIDIA GTX960 GPU.

For further exploration of related research in these domains, interested readers are encouraged to consult the following sources: Refs. [46,47] for traffic sign detection, Refs. [48,49] for traffic light detection, and Ref. [50] for road surface marking detection.

4. Proposed Embedded System

4.1. Research Objective and Goals

The primary objective of this study was to devise a prototype system capable of detecting pedestrians, discerning their movements, and accurately interpreting road signs with a target accuracy of 90%. The developed system was designed to operate effectively under various challenging conditions, including nighttime, foggy weather, and rainy conditions.

The official frame rate in ADASs ranges from 5 to 60 FPS (frames per second). However, this depends on the specific objectives of the cameras employed. In practical applications, systems often necessitate a higher frame rate to effectively monitor rapid changes in the distance, but for detecting traffic signs a lower frame rate of 10 FPS is deemed sufficient.

4.2. Tools Used in the Study

We worked with a NVIDIA Jetson Nano B01 4 GB developer kit, a specialized platform tailored for AI applications and image-centric solutions [51]. The device features a 4-core ARM A57 @ 1.43 GHz CPU, 4 GB 64-bit LPDDR4 RAM, and a 128-core Maxwell GPU (Manufacturer: NVIDIA; Country of origin: Santa Clara, CA, USA), rendering it an optimal choice for the task at hand. Additionally, the compact size of the device allows convenient installation within a car, with power supplied through a 5V 4A cigarette lighter adapter. Furthermore, the B01 version of the device is equipped with two MIPI CSI-2 inputs, allowing the simultaneous connection of two cameras.

Recognizing the challenge posed by reduced detection accuracy in low-light conditions, as demonstrated in prior studies [52,53], the system strategically integrates two CSI cameras: the PI v2.1 [54] for daytime operations and the IMX219-160IR [55] for nighttime scenarios. The preference for CSI cameras over USB alternatives is attributed to their superior bandwidth [56,57], with the capability to achieve up to 120 FPS at a resolution of 1280×720.

We designed a special camera housing using a 3D printer to accommodate both the cameras and the NVIDIA Jetson. This specialized housing was intricately designed for attachment to the rear-view mirror, featuring a joint mechanism that enabled independent adjustment of the cameras to attain the optimal viewing direction. To facilitate user interaction, the system incorporated a 7″ touchscreen LCD display, as illustrated in Figure 1.

Figure 1. Complete setup installed in the car.

4.3. System Working Principle

Figures 2 and 3 depict the procedural steps of the computer vision and deep learning techniques employed in the developed system. The system utilized two cameras: the

primary PI V2.1 camera and the secondary IMX219-160IR (Manufacturer: Waveshare; Country of origin: Shenzhen, China) camera.

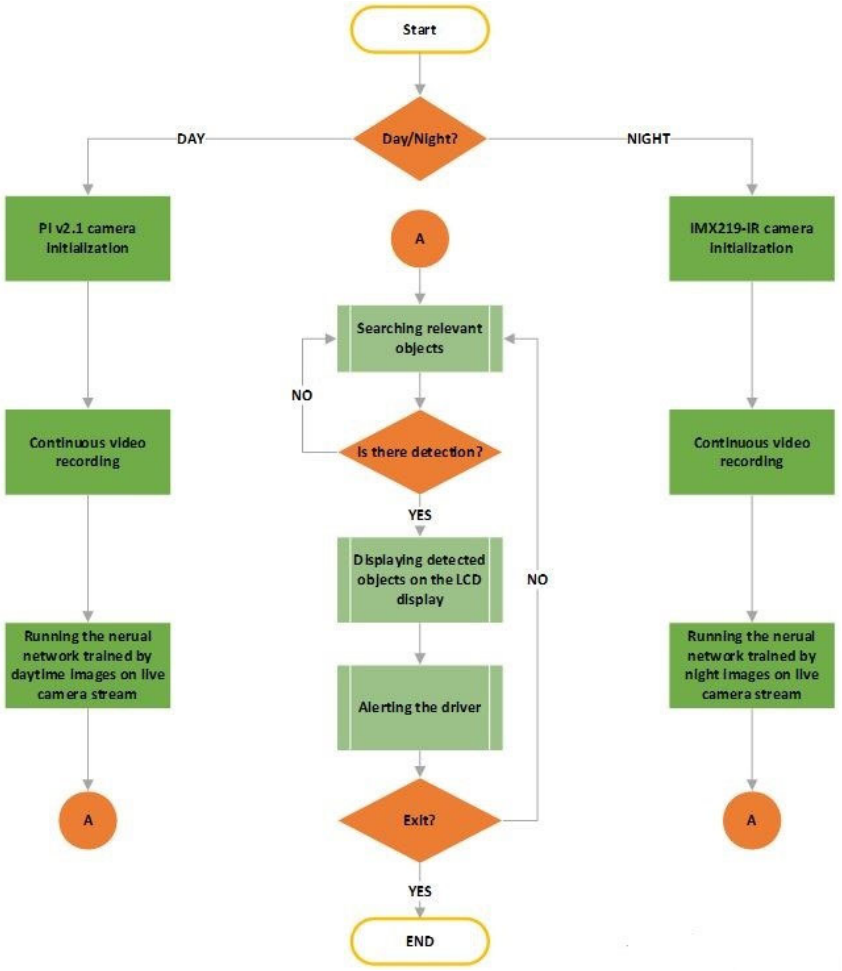

Figure 2. Flowchart of the working principle of the planned system.

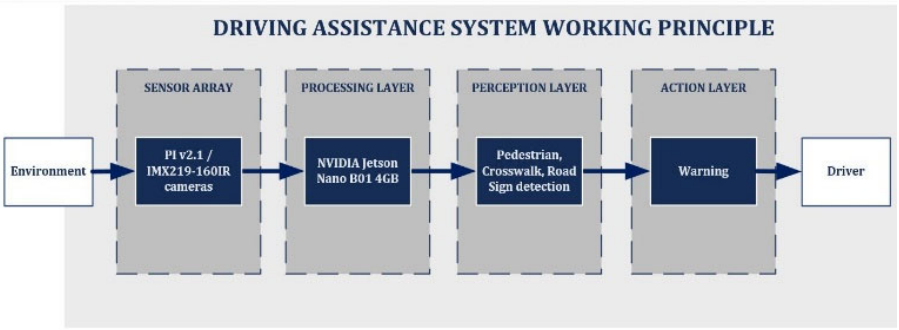

Figure 3. Block diagram of the planned system.

The system incorporated an LCD display to promptly notify the driver of their obligation to yield priority. This display plays a pivotal role, offering real-time feedback and information about detected objects in the road environment. Using deep learning techniques and advanced neural networks ensures high accuracy and sensitivity, establishing the system's reliability and effectiveness in ensuring the safety of both drivers and pedestrians.

5. Experimental Setup

5.1. Preparation of the Training Data Set

The initial and crucial step in this study involved the assembly of a representative image database, mirroring real-world scenarios. This step was imperative for the effective training of the neural network, ensuring its competence in real traffic conditions. Given the system's intended operation amidst actual traffic, we sourced images from authentic environments. Despite the abundance of datasets related to traffic signs and pedestrians, only relevant ones were utilized—specifically, those focusing on traffic signs indicative of priority, given the potentially severe consequences for both drivers and pedestrians.

To bolster the system's reliability in real-time use, a proprietary dataset was meticulously crafted, aligning the positions and orientations of traffic signs and pedestrians in images with their live camera stream counterparts. We sourced images directly from the same cameras implemented into the system providing a more authentic representation of the system's anticipated operating environment. The main goal was to enhance the system's dependability across diverse conditions encountered in daily life. Such conditions encompass variations in weather, including cloudy/rainy weather or instances during sunrise/sunset, as well as scenarios with partially obscured objects, such as tree branches (see Figure 4). The dataset comprised over 740 images captured by the two cameras, encompassing 400 daytime images and 340 night images, spanning diverse weather and sunlight conditions (see Figure 5).

Figure 4. Partially rotated/unfolded traffic signs.

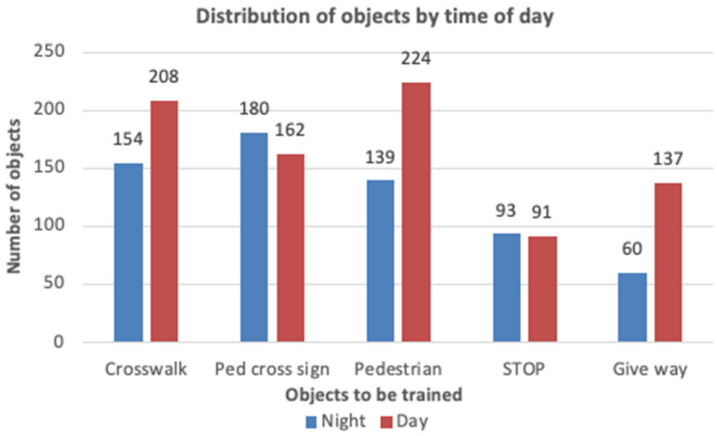

Figure 5. Distribution of objects by time of day.

We undertook the standardization of the collected images to a consistent size of 1280 × 720 and annotated the objects in them using the CVAT (Computer Vision Annotation Tool). The resulting annotations encompassed categories and coordinates of the objects, meticulously stored in XML files. The dataset underwent a systematic partitioning into three distinct subsets: 70% of the images designated for training, 15% for validation, and another 15% for testing purposes. The validation subset served as a crucial gauge for evaluating the model's performance post each epoch, while the testing subset encompassed images that had not been encountered by the learning algorithm during training.

5.2. Selection of the Pre-Trained Neural Network

The work presented in [58] provides a comprehensive overview of challenges associated with pre-trained models. Neural network layers are distinctly assigned specific responsibilities: the early layers encapsulate general properties, while the middle layers focus on identifying shapes or task-specific characteristics. Transfer learning introduces a distinctive feature wherein only the end of the pre-trained network is retrained, while the other layers remain frozen. This approach enjoys widespread adoption in deep learning, owing to its advantageous features, including accelerated training times, enhanced performance, and a relatively modest data requirement.

When selecting a neural network for a specific application, the key consideration is not identifying the optimal model, but rather determining the architecture and configuration that strikes the best balance between accuracy and speed [59].

The "SSD w/MobileNet, Lo Res" model offers excellent speed coupled with a moderate average precision compared to alternative models. This particular model, characterized as a Single Shot Detection Convolutional Neural Network, excels in detecting multiple objects within a single image with the added benefits of low power consumption and simplified computations, rendering it apt for deployment in embedded devices. Given the real-time operational requirement of the planned system, prioritizing speed was of paramount importance, leading to the deliberate selection of this neural network.

5.3. Training Stage

The Jetson Nano has been equipped with a range of SDKs and toolkits, including JetPack 4.6.1, TensorRT 8.2.1, CUDA Toolkit 10.2, cuDNN 8.2.1, and PyTorch 1.1.0. We chose PyTorch due to its simplicity and growing popularity in recent years. The "jetson-inference" library was used to train, optimize, and run the model on the Jetson hardware, eliminating the necessity for a robust computer with a GPU for tasks such as model training, conversion, and optimization. This approach not only conserves time but also ensures the efficient utilization of the Jetson Nano's hardware capabilities when running within the TensorRT framework. We used Python codes from the "jetson-inference" library [60] for training, evaluation, optimization, and inference of the "mo-bilenet-v1-ssd-mp-0_675.pth" model on both the test images and the live camera stream using the detectNet module. This architecture processes 300 × 300 images as input.

We implemented two learning schedules for model training. The initial approach adopted a stepwise reduction of the learning rate (Step Decay/Multi-Step Learning Rate), starting at 10-3 and decreasing it by a factor of 10 every 50 steps (as per Formula (2)). The second method involved testing a technique wherein the learning rate continuously diminished following the cosine function (Formula (3)). The training process spanned 120 epochs in both instances.

$$LR_i = \begin{cases} 10^{-3}, & T_i < 50 \\ 10^{-4}, & 50 \leq T_i < 100 \\ 10^{-5}, & 100 \leq T_i \leq 120 \end{cases}, \quad (2)$$

- LR_i current learning rate;
- T_i: current epoch.

$$LR_i = \frac{1}{2} * LR_{max} * \left(1 + (\cos \frac{T_i}{T_n} * \pi)\right), \quad (3)$$

- LR_{max}: initial learning rate (0.01);
- T_n: all epochs (120).

5.4. Evaluation Stage and Metrics Applied

In computer vision, precision and recall serve as prevalent evaluation metrics for object detection [61]. Precision signifies the accuracy of predictions (as per Formula (4)), while recall characterizes the percentage of objects detected by the model (as per Formula (5)).

During evaluation, accuracy values are computed the using IoU [62] threshold, indicating the extent of overlap between the predicted and ground-truth bounding boxes. Using this value, the algorithm assesses the confidence value of the detected object (as per Formula (6)).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{5}$$

- TP: true positive: correct predictions;
- FP: false positive: incorrect predictions;
- FN: false negative: number of unidentified objects.

$$\text{Confidence} = \Pr(\text{Object}) * \text{IoU}(\text{truth}, \text{pred}), \tag{6}$$

- Pr (Object): binary value which is equal to 1 if the object is present in the specified cell, and 0 otherwise;

After the training phase, we conducted a validation process by evaluating every tenth checkpoint file to determine the models' average precision values. The models with the highest precision values were selected. Among the various learning rate techniques tested, the cosine annealing method yielded the most favorable outcomes. The accuracy values reached 92.25% at epoch 89 (day) and 95.15% at epoch 112 (night). We exported the resulting checkpoint files to ONNX (Open Neural Network Exchange) format for deployment with detectNet. Figures 6 and 7 illustrate the relationship between the loss function and accuracy values across epochs, along with the associated learning rate.

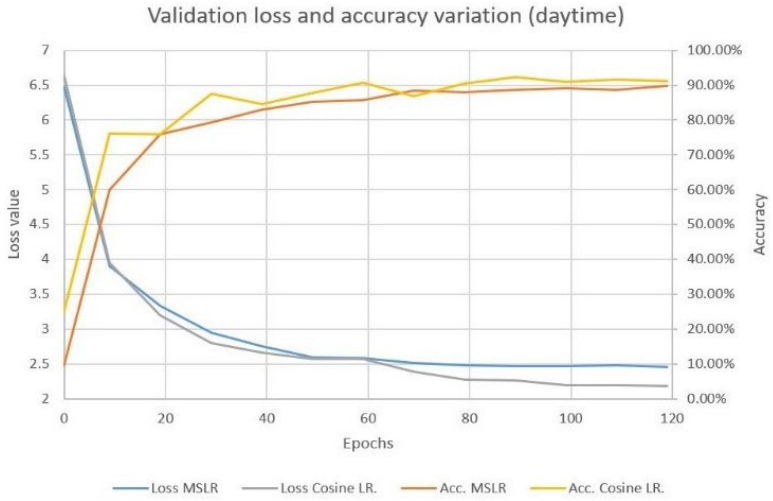

Figure 6. Comparing validation loss and accuracy variation for the daytime model.

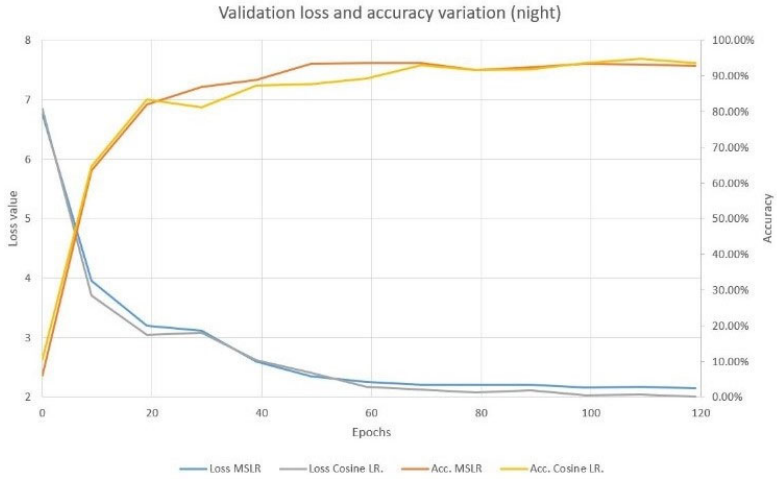

Figure 7. Comparing validation loss and accuracy variation for the nighttime model.

Since the "jetson-inference" library calculates only the average precision of the model on each checkpoint file, the recall metric cannot be determined. Thus, after exporting the networks, we evaluated the checkpoint files on the test dataset images to compute the average recall value with Excel and Formula (5). (Predictions with a confidence value of at least 50% were considered true): The average recall values were 50% (day) and 48.87% (night).

Throughout the validation process, it became evident that employing low-resolution input images (300 × 300) led to a substantial information loss, particularly affecting the detection of smaller objects like road signs from a distance (Figure 8). To mitigate this challenge, we opted for a neural network capable of handling larger input images. As the "jetson-inference" library lacks support for models designed for larger input images, an alternative approach was pursued to address this limitation.

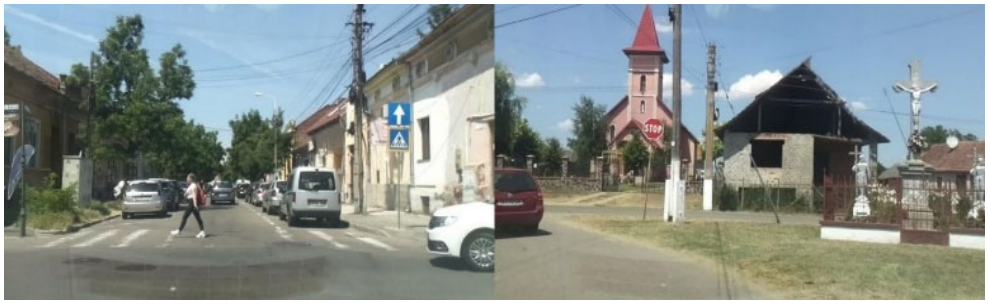

Figure 8. Unidentified objects.

5.5. Training of SSD MobileNet 640 × 640 FPNLite

The "SSD MobileNet 640 × 640 FPNLite" pre-trained model was obtained from the Tensorflow 2 Model Zoo [63] and underwent retraining on a personal computer equipped with an Intel(R) Core(TM) i7-8750H CPU 2.20 GHz, 8 GB RAM, and an NVIDIA Geforce GTX 1060 6 GB video card (Manufacturer: ASUS; Country of origin: Bucharest, Romania). We updated the video card driver on the PC and installed the CUDA (11.2) and cuDNN (8.1) drivers [64] compatible with the Tensorflow version (2.7.0) to enable the learning process to run on the GPU for better efficiency. We created a virtual environment in Anaconda Prompt where the Tensorflow framework, Object Detection API, and necessary modules were installed [65].

The dataset used in the previous chapter served as the foundation for training the model, with an 80% split for training and 20% for testing. The training process consisted of 25,000 iterations, with a batch size of 4 and a learning rate specified in Figure 9. The model's overall loss function encompassing regularization error, localization error, and classification error, was minimized to 0.074 within a duration of 2.5 h.

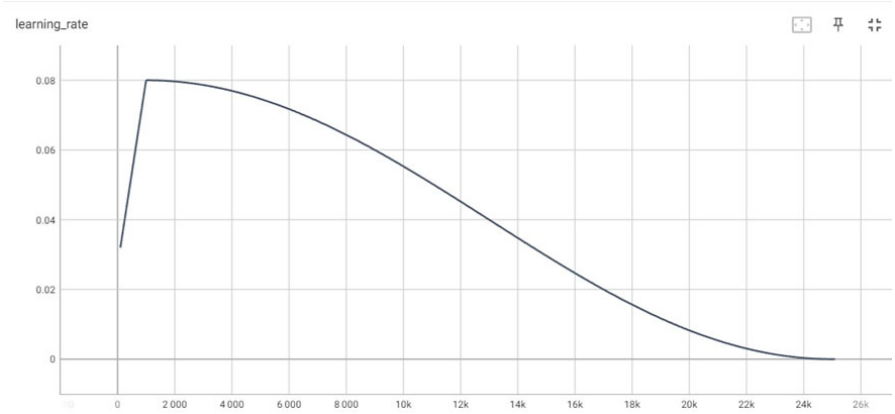

Figure 9. Applied learning rate during the training process.

Following the conclusion of the training process, we exported the model using the script "exporter_main_v2.py" and subsequently validated using the script "model_main_tf2.py". During the testing phase, it became apparent that the model exhibited lower performance on the test dataset, manifesting some false predictions, despite the effective minimization of the loss function during training. This phenomenon, where the model excels on the training dataset but struggles with the test dataset, is known as overfitting [66]. The overfitted model was highly tailored to the training dataset and failed to generalize to new data, resulting in erroneous outcomes and suboptimal decisions.

5.6. Optimizing Model Performance with Data Augmentation

Increasing the number of images in a training dataset is an effective strategy to address overfitting, although collecting new data can be challenging and time-consuming. Data augmentation [67] is a useful technique to help learning algorithms detect significant features, particularly when training data is insufficient or lacks diversity. Augmentation involves generating modified versions of existing images, such as rotation, mirroring, or changes in color space, without the need for additional human resources to label new images. Additional augmentation methods include zooming, shifting, adding noise, and adjusting brightness, saturation, and contrast values. Careful consideration of the degree of augmentation is essential to ensure that the image content retains its meaningfulness for robust generalization.

This study employed the following data augmentation techniques: random cropping with a probability of 87.5%, random brightness adjustment with a maximum of 20%, contrast adjustment within a range of ±20% (Figure 10), saturation adjustment within a range of ±20%, and hue adjustment with a maximum of 1%.

Figure 10. Contrast change: −20%; 0%; +20% (author's own figure).

Regarding the variation of the loss function, it was apparent that the loss function of the current iteration was less than optimal compared to its previous counterpart. As shown in Figure 11, the current loss function displayed spikes, and numerous outlier values, as well as minor divergences. This phenomenon was attributed to the model's inability to make precise predictions on the augmented images, resulting in an increased error rate. Subsequent evaluation of the model revealed a substantial improvement in accuracy (Figure 12) compared to the previous training cycle.

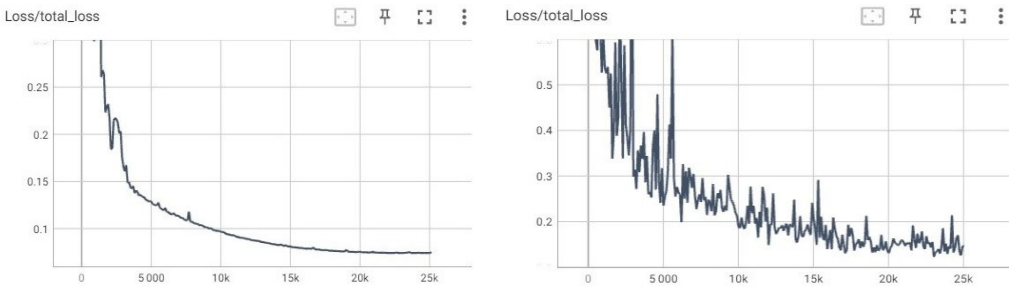

Figure 11. Loss comparison: before and after data augmentation.

Figure 12. Comparison: before and after data augmentation.

5.7. Custom Anchor Boxes for Improved Object Detection

Another common issue encountered during the study was related to the model's proficiency in recognizing crosswalks, which were frequently overlooked. Upon examining the deep learning parameters in the "pipeline.config" file, it became evident that the default aspect ratios of anchor boxes (0.5, 1.0, and 2.0) were insufficient to encompass most crosswalks. To address this, we calculated the aspect ratios of the objects in the entire dataset; Figure 13 illustrates a histogram indicating that the majority of objects in the entire dataset fell within the first three categories: pedestrians (0.5), traffic signs (1 and 1.25), while the pedestrian crossing signs were distributed across several categories. Defining a large number of scales for the algorithm would inevitably slow down both the model training and its execution.

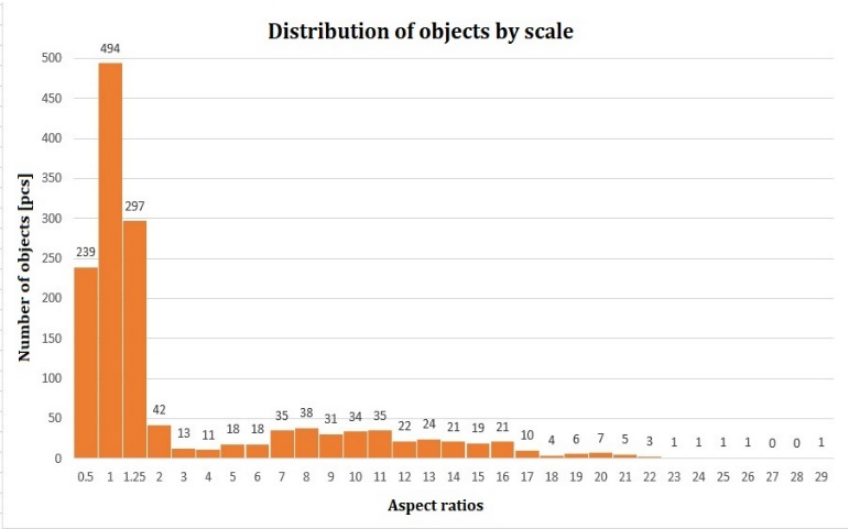

Figure 13. Distribution of objects by aspect ratios.

To address this issue, we defined supplementary intervals, amounting to six in total, and retrained the model with parameters adjusted accordingly to accommodate these intervals (Figure 14).

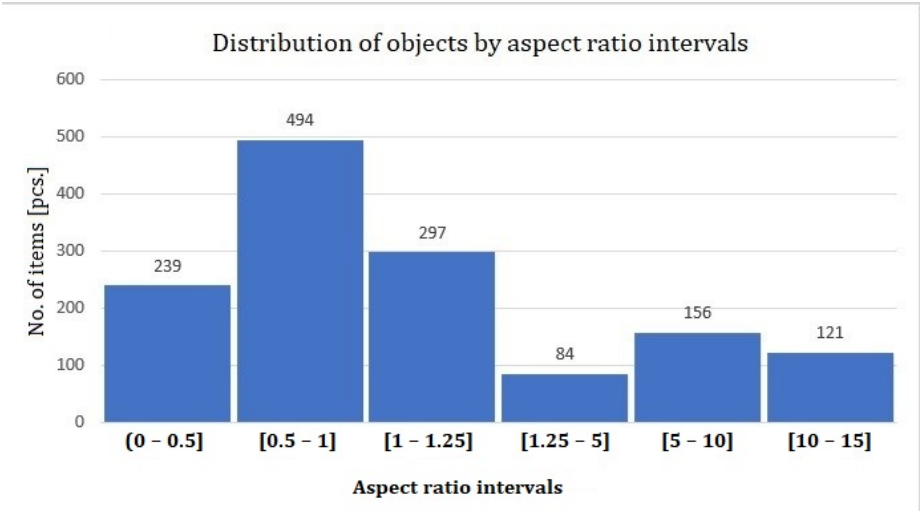

Figure 14. Defining new anchor box aspect ratios for training.

After rerunning the learning algorithm, we obtained a refined model (as demonstrated in Figure 16), that underwent subsequent validation to verify its improved performance.

We evaluated the model once again on the test images with an IoU threshold of 0.5. Although the numerical results (Figure 15) might not have fully captured it, a significant improvement was observed compared to the previous state of the model, as depicted in Figure 16. The left side represents the previous model state loaded with the default anchor box scales, while the right side illustrates the enhanced performance achieved with the scales defined in this study. Moreover, this approach successfully addressed the problem of the model overlooking pedestrian crossing markings.

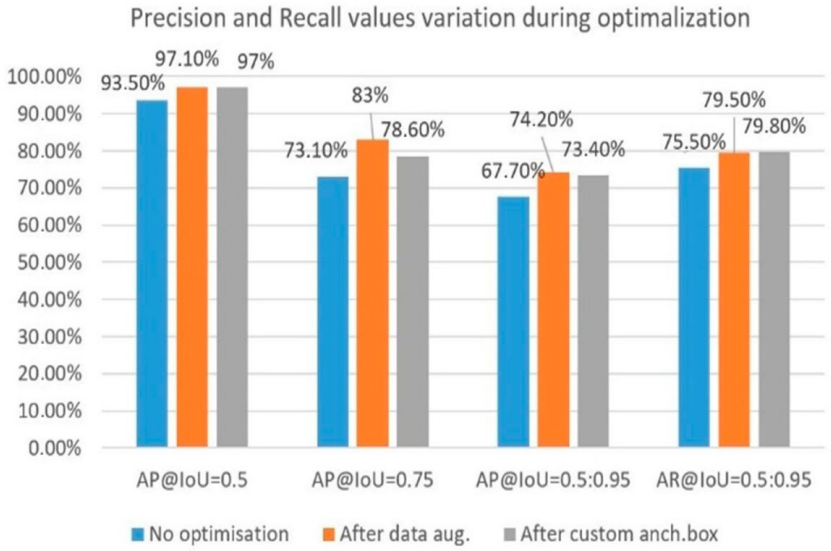

Figure 15. Comparing of validation results for different optimization methods.

Figure 16. Comparison: Before vs. after using custom anchor boxes.

It is natural to question how improvement could be observed despite the absence of improvement in the chart values. The observed improvement could be attributed to the increase in localization error (by 1.17%), leading to a decrease in the model's average precision and sensitivity at higher IoU thresholds. As the recall value was averaged over various IoU thresholds (0.50, 0.55, 0.6, . . ., 0.95) by the code, it is probable that the model's sensitivity had improved for the lower IoU values and degraded for the higher ones. This claim is supported by the fact that for an IoU of 0.75, the accuracy of the neural network declined by 4.4%.

In the proposed system, the precise localization and framing of objects by the algorithm is not a critical aspect. The primary objective is to accurately classify the objects into their respective categories and avoid any incorrect predictions or neglect. Consequently, the benchmark for the system's performance is based on the achieved characteristics at a coverage threshold of 0.50, which is considered a successful outcome.

5.8. Tensorflow Model Optimization for TensorRT

The real-time execution of deep learning neural networks on edge devices is a crucial factor for enabling many application areas. However, given the limited memory, computing resources, and power of these devices, it is necessary to optimize these networks for embedded applications. The subsequent stage of this project focused on transforming the embedded model into a format that is compatible with TensorRT. This was achieved by exporting the Tensorflow file with the ".pb" extension to the ".ONNX" format, followed by creating an ".engine" file from this format that could be executed on the Jetson Nano, with the TensorRT framework. For this process, we applied the Python codes in the "TensorRT" library [68] on NVIDIA's Github website.

TensorRT provides support for three types of precision: single precision (FP32), half precision floating point (FP16), and eight-bit signed integer (INT8). The application of INT8 quantization has gained popularity in several machine learning frameworks as it generates a four-fold reduction in data storage requirements, thus reducing the memory

necessary to store all the weights and activations in the neural network. To achieve the eight-bit representation of the floating-point tensor (FP32 or float32) (Formula (7)), a scale factor (Formula (8)) is used, mapping the dynamic range of the tensor to the [−128; 127] interval [69].

$$x_q = \text{Clip}\left(\text{Round}\left(\frac{x_f}{\text{scale}}\right)\right), \tag{7}$$

- x_f :Single-precision floating-point (FP32) tensor;
- x_q: INT8 representation of FP32 tensor;
- Round(): A function that rounds rational numbers to integers;
- Clip(): A function that clips outliers that fall outside the [−128; 127] interval.

$$\text{scale} = (2 * a_{max})/256, \tag{8}$$

$$a_{max} = \max(\text{abs}(x_f)) \tag{9}$$

To ensure no degradation in performance, the FP32 model is evaluated with a small dataset representative of the task's real input data, and statistics on the interlayer activation distributions are collected. This process is "calibration" [69] and the representative dataset used is the "calibration dataset".

In this study, the calibration dataset comprised 320 images captured during the daytime and 272 images captured during the nighttime. To evaluate the model's precision and recall for IoU = 0.5, we utilized the same test dataset as in the previous sections. The results indicated an average precision of 98.45% for daytime images and 100% for nighttime images, and an average recall of 87.59% for daytime images and 90.68% for nighttime images.

After the optimization process using TensorRT (8.2.1), we observed an improvement in the sensitivity during validation on the test images as compared to the results obtained previously, as demonstrated in Figure 17.

Figure 17. Comparison: Before vs. after optimization to TensorRT.

The use of higher-resolution input images significantly mitigated information loss, resulting in enhanced detection capabilities for smaller objects including a distant road sign. The implementation of data augmentation and customized anchor boxes led to improvements in both average precision (AP) and average recall (AR) metrics, surpassing the values obtained with the previous architecture, as illustrated in Figure 18.

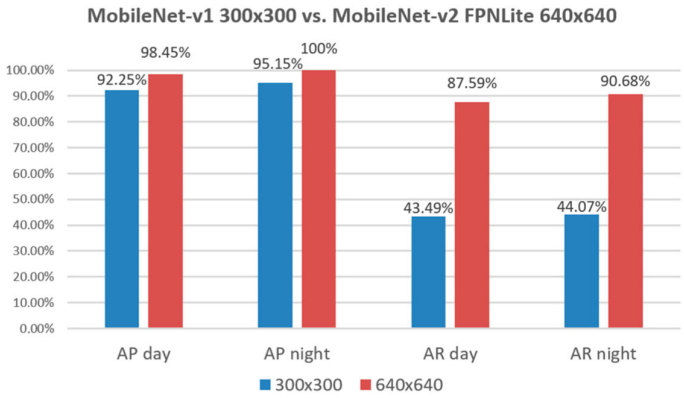

Figure 18. Comparison: SSD MobileNet 300 × 300 vs. 640 × 640.

6. Testing Stage

After evaluating the model on the test dataset, it was deployed on the Jetson Nano, processing live-camera footage through the DeepStream SDK at a runtime of approximately 8.7 FPS. During execution, the model predicts the probability of objects within the frames and displays those surpassing a confidence value threshold of 50%, as specified in Formula (6).

We conducted comprehensive testing across diverse lighting conditions, including daytime, sunrise, sunset, dusk, night, and rainy environments. Evaluations were extended to various locations, spanning different cities and villages, to thoroughly assess the model's adaptability, performance, and limitations.

6.1. Daytime Testing

During real-time testing, the system demonstrated an overall 98.45% accuracy in detecting both pedestrians and traffic signs, with 87.59% sensitivity in detecting them irrespective of their angles or orientations, as illustrated in Figure 19.

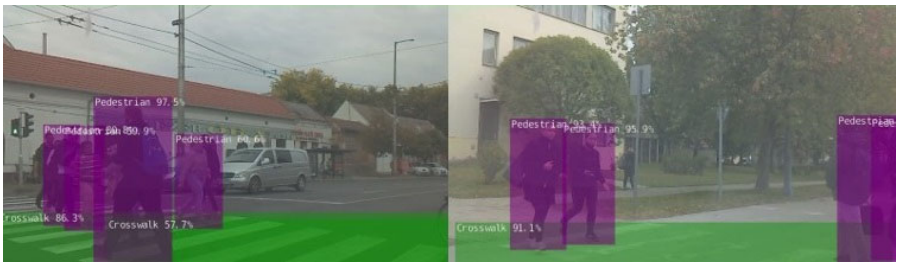

Figure 19. Testing pedestrian detection in daytime.

We tested the system during both dawn (Figure 20) and sunset (Figure 21). It played a substantial role in assisting the driver, particularly in situations where the sun could potentially blind the driver. Challenges were encountered, especially with pedestrian crossing road markings, where reflective light more frequently interfered with the system.

Figure 20. Testing at sunrise.

Figure 21. Testing in the afternoon, at sunset.

6.2. Nocturnal Testing

In situations of low-light conditions like twilight, the system experienced a decrease in reliability. Nevertheless, this issue could be mitigated by incorporating the IR camera, coupled with a model specifically trained on images captured during nighttime as shown in Figures 22 and 23.

Figure 22. Testing in twilight.

Figure 23. Testing pedestrian detection in twilight.

During testing in low-light conditions such as night, the model exhibited an overall 100% detection accuracy for both pedestrians and traffic signs, with 90.68% sensitivity in detecting them. Additionally, we tested the system in rainy weather conditions at night, and despite the challenging circumstances, it accurately detected pedestrians and road signs in a dark, unlit street with dimmed headlights and through a rainy windscreen (as illustrated in Figure 24).

Figure 24. Testing at night, in rain.

7. Conclusions

In conclusion, this paper introduces an embedded system designed for real-time pedestrian and priority sign detection, addressing the limitations of the existing ADAS in terms of cost-effectiveness and universal applicability. The proposed solution, incorporating two cameras, an NVIDIA Jetson Nano B01 edge device and an LCD display, provides an affordable alternative applicable to any vehicle.

This research distinguishes itself by simultaneously addressing priority traffic sign recognition and pedestrian detection, with a specific focus on the most critical objects: pedestrians, pedestrian crossings, stop signs, and give way signs. Moreover, the experiment involved training the applied neural network with our own database; it is crucial to acknowledge that the evaluation results presented here are based on a relatively modest dataset of 320 images.

While the system demonstrated promising average accuracy and recall rates surpassing 90%, coupled with an inference speed of 8.7 frames per second, slightly below the predetermined target, it is crucial to approach definitive conclusions with caution due to the constraints of the limited dataset. Real-time testing in various lighting conditions and driving scenarios provided preliminary insights into the system's early effectiveness.

Considering its average accuracy and recall rates, alongside a modest inference speed, our proposed system stands as a viable option for deployment in urban traffic scenarios.

The cost-effectiveness and improved performance of the system position this study as foundational groundwork for enhancing the safety and accessibility of autonomous driving technologies. Future research endeavors will focus on expanding both the training and test datasets, conducting more extensive testing to further validate and refine the system's capabilities.

8. Further Development Possibilities

To operate the current system, a keyboard and mouse are currently required for the driver to manually select the desired mode based on their judgement. An innovative improvement involves the integration of light sensors, enabling dynamic assessments of light conditions for real-time adjustments to the relevant model. Additionally, we are investigating an automated feature capable of identifying human activities associated with adjusting to lighting conditions, such as activating headlights, which presents a promising avenue for advanced automation.

The prospect of integrating traffic lights into the system provides an opportunity to address challenges at signalized intersections, particularly instances of simultaneous green light and stop sign signals. However, this endeavor introduces inherent complexities, necessitating the implementation of a lane recognition algorithm to ensure that the system selectively monitors the traffic light corresponding to the user's lane.

Furthermore, enhancing the neural network's training dataset through the inclusion of additional images represents a pivotal improvement. This augmentation extends the system's applicability across a broader spectrum of scenarios and holds considerable potential to elevate the overall performance and robustness of the system in diverse real-world scenarios.

Author Contributions: Conceptualization, K.S., L.A. and J.M.; methodology, K.S. and L.A.; formal analysis, K.S. and L.A.; investigation, K.S. and L.A.; resources K.S., L.A. and J.M.; data curation, K.S. and L.A.; writing—original draft preparation, K.S., L.A. and J.M.; writing—review and editing, K.S., L.A. and J.M.; visualization, K.S. and L.A.; project administration, J.M.; funding acquisition, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by University of Debrecen, Faculty of Engineering.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Petridou, E.; Moustaki, M. Human Factors in the Causation of Road Traffic Crashes. *Eur. J. Epidemiol.* **2000**, *16*, 819–826. [CrossRef]
2. World Health Organization. 20 June 2022. Available online: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (accessed on 25 March 2023).
3. Lu, M.; Wevers, K.; Van Der Heijden, R. Technical Feasibility of Advanced Driver Assistance Systems (ADAS) for Road Traffic Safety. *Transp. Plan. Technol.* **2005**, *28*, 167–187. [CrossRef]
4. Farhat, W.; Sghaier, S.; Faiedh, H.; Souani, C. Design of efficient embedded system for road sign recognition. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *10*, 491–507. [CrossRef]
5. Jian, L.; Li, Z.; Yang, X.; Wu, W.; Ahmad, A.; Jeon, G. Combining Unmanned Aerial Vehicles with Artificial-Intelligence Technology for Traffic-Congestion Recognition: Electronic Eyes in the Skies to Spot Clogged Roads. *IEEE Consum. Electron. Mag.* **2019**, *8*, 81–86. [CrossRef]
6. Li, Q.; Wu, W.; Lu, L.; Li, Z.; Ahmad, A.; Jeon, G. Infrared and visible images fusion by using sparse representation and guided filter. *J. Intell. Transp. Syst.* **2019**, *24*, 254–263. [CrossRef]
7. Levinson, J.; Askeland, J.; Becker, J.; Dolson, J.; Held, D.; Kammel, S.; Kolter, J.Z.; Langer, D.; Pink, O.; Pratt, V.; et al. Towards fully autonomous driving: Systems and algorithms. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden, Germany, 5–9 June 2011; pp. 163–168.

8. Saadna, Y.; Behloul, A. An overview of traffic sign detection and classification methods. *Int. J. Multimedia Inf. Retr.* **2017**, *6*, 193–210. [CrossRef]
9. Abdulrahim, K.; Al-Smadi, M.; Salam, R.A. Traffic Surveillance: A Review of Vision Based Vehicle Detection, Recognition and Tracking. *Int. J. Appl. Eng. Res.* **2016**, *11*, 713–726.
10. Prakash, C.D.; Akhbari, F.; Karam, L.J. Robust obstacle detection for advanced driver assistance systems using distortions of inverse perspective mapping of a monocular camera. *Robot. Auton. Syst.* **2018**, *114*, 172–186. [CrossRef]
11. Rajaram, R.N.; Ohn-Bar, E.; Trivedi, M.M. RefineNet: Refining Object Detectors for Autonomous Driving. *IEEE Trans. Intell. Veh.* **2017**, *1*, 358–368. [CrossRef]
12. Brunetti, A.; Buongiorno, D.; Trotta, G.F.; Bevilacqua, V. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing* **2018**, *300*, 17–33. [CrossRef]
13. Ullah, A.; Muhammad, K.; Del Ser, J.; Baik, S.W.; de Albuquerque, V.H.C. Activity Recognition Using Temporal Optical Flow Convolutional Features and Multilayer LSTM. *IEEE Trans. Ind. Electron.* **2018**, *66*, 9692–9702. [CrossRef]
14. Intel. 2014. Available online: https://les-svc.org/resources/LES-SVC%205-18-16%20Article%20-%20Intel-automotive-autonomous-driving-vision-paper.pdf (accessed on 1 February 2024).
15. Tesla. Available online: https://www.tesla.com/autopilot (accessed on 15 March 2023).
16. Mobileye. Available online: https://www.mobileye.com/true-redundancy/ (accessed on 15 March 2023).
17. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
18. Turay, T.; Vladimirova, T. Toward Performing Image Classification and Object Detection with Convolutional Neural Networks in Autonomous Driving Systems: A Survey. *IEEE Access* **2022**, *10*, 14076–14119. [CrossRef]
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
20. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; Volume 28, pp. 91–99.
21. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, C.A. SSD: Single Shot MultiBox Detector. In *Computer Vision–ECCV 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
22. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
23. Quassim, H.; Verma, A.; Feinzimer, D. Compressed residual-VGG16 CNN model for big data places image recognition. In Proceedings of the IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018.
24. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichencko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision. *arXiv* **2017**, arXiv:1704-04861.
25. Chiu, Y.-C.; Tsai, C.-Y.; Ruan, M.-D.; Shen, G.-Y.; Lee, T.-T. Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems. In Proceedings of the International Conference on System Science and Engineering (ICSSE), Kagawa, Japan, 31 August–3 September 2020; pp. 1–5.
26. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
27. McKinney, W. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; Volume 445, pp. 51–56.
28. McKinney, W. Pandas: Powerful Python Data Analysis Toolkit. Available online: https://pandas.pydata.org/pandas-docs/version/0.25.3/pandas.pdf (accessed on 1 February 2024).
29. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
30. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. TensorFlow: A system for large-scale machine learning. *arXiv* **2016**, arXiv:1605.08695.
31. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L. PyTorch: An Imperative Style, High-Performance. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, New York, NY, USA, 8 December 2019; pp. 8026–8037.
32. Papers With Code. Available online: https://paperswithcode.com/trends (accessed on 25 March 2023).
33. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional Architecture for Fast Feature Embedding. In Proceedings of the 22nd ACM international conference on Multimedia, Orlando, FL, USA, 7–8 November 2014; pp. 675–678.
34. Chollet, F. Keras: Deep Learning for humans. Available online: https://keras.io/keras_3/ (accessed on 1 February 2024).
35. Aljaafreh, A. Camera-Based Driver Monitoring System for Abnormal Behavior Detection. *Jordan J. Electr. Eng.* **2020**, *6*, 205–215. [CrossRef]
36. Oczan, K.; Sharma, A.; Knickerbocker, S.; Merickel, J.; Hawkins, N.; Rizzo, M. Road Weather Condition Estimation Using Fixed and Mobile Based Cameras. In *Advances in Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 192–204.
37. Ye, X.-Y.; Hong, D.-S.; Chen, H.-H.; Hsiao, P.-Y.; Fu, L.-C. A two-stage real-time YOLOv2-based road marking detector with lightweight spatial transformation-invariant classification. *Image Vis. Comput.* **2020**, *102*, 103978. [CrossRef]

38. Barba-Guaman, L.; Naranjo, J.E.; Ortiz, A. Deep Learning Framework for Vehicle and Pedestrian Detection in Rural Roads on an Embedded GPU. *Electronics* **2020**, *9*, 589. [CrossRef]
39. Hechri, A.; Mtibaa, A. Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks. *IET Image Process.* **2020**, *14*, 939–946. [CrossRef]
40. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005; pp. 886–893. [CrossRef]
41. Wali, S.B.; Hannan, M.A.; Hussain, A.; Samad, S.A. An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, Shape Matching, and SVM. *Math. Probl. Eng.* **2015**, *2015*, 250461. [CrossRef]
42. Muller, J.; Dietmayer, K. Detecting Traffic Lights by Single Shot Detection. In Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 266–273.
43. Fregin, A.; Muller, J.; Krebel, U.; Dietmayer, K. The DriveU Traffic Light Dataset: Introduction and Comparison with Existing Datasets. In Proceedings of the 2018 IEEE international conference on robotics and automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 3376–3383.
44. Guney, E.; Bayilmis, C.; Cakan, B. An Implementation of Real-Time Traffic Signs and Road Objects Detection Based on Mobile GPU Platforms. *IEEE Access* **2022**, *10*, 86191–86203. [CrossRef]
45. Ayachi, R.; Afif, M.; Said, Y.; Ben Abdelali, A. Traffic Sign Recognition Based on Scaled Convolutional Neural Network for Advanced Driver Assistance System. In Proceedings of the IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS), Genova, Italy, 9–11 December 2020; pp. 149–154.
46. Wu, Y.; Li, Z.; Chen, Y.; Nai, K.; Yuan, J. Real-time traffic sign detection and classification towards real traffic scene. *Multimedia Tools Appl.* **2020**, *79*, 18201–18219. [CrossRef]
47. Chiu, Y.-C.; Lin, H.-Y.; Tai, W.-L. Implementation and Evaluation of CNN Based Traffic Sign Detection with Different Resolutions. In Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Taipei, Taiwan, 3–6 December 2019; pp. 1–2.
48. Weber, M.; Huber, M.; Zollner, J.M. HDTLR: A CNN based Hierarchical Detector for Traffic Lights. In Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 255–260.
49. Ouyang, Z.; Niu, J.; Liu, Y.; Guizani, M. Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles. *IEEE Trans. Mob. Comput.* **2019**, *19*, 300–313. [CrossRef]
50. Hoang, T.M.; Nam, S.H.; Park, K.R. Enhanced Detection and Recognition of Road Markings Based on Adaptive Region of Interest and Deep Learning. *IEEE Access* **2019**, *7*, 109817–109832. [CrossRef]
51. NVIDIA. Available online: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed on 25 March 2023).
52. Nawrat, A.; Daniec, K.; Warmuz, T. Object Detection Using IR Camera. In *Advanced Technologies for Intelligent Systems of National Border Security*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 440, pp. 129–142.
53. Park, J.; Chen, J.; Cho, Y.K.; Kang, D.Y.; Son, B.J. CNN-Based Person Detection Using Infrared Images for Night-Time Intrusion Warning Systems. *Sensors* **2019**, *20*, 34. [CrossRef] [PubMed]
54. Raspberry PI. Available online: https://www.raspberrypi.com/products/camera-module-v2/ (accessed on 25 March 2023).
55. Waveshare Electronics. Available online: https://www.waveshare.com/wiki/IMX219-160IR_Camera (accessed on 25 March 2023).
56. Kumar, P. e-con Systems. 11 November 2021. Available online: https://www.e-consystems.com/blog/camera/technology/mipi-camera-vs-usb-camera-a-detailed-comparison/ (accessed on 25 March 2023).
57. MIPI Alliance. Available online: https://www.mipi.org/driving-the-wires-of-automotive (accessed on 25 March 2023).
58. Tao, W.; Al-Amin, M.; Chen, H.; Leu, M.C.; Yin, Z.; Qin, R. Real-Time Assembly Operation Recognition with Fog Computing and Transfer Learning for Human-Centered Intelligent Manufacturing. *Procedia Manuf.* **2020**, *48*, 926–931. [CrossRef]
59. Hui, J. Medium. 28 March 2018. Available online: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (accessed on 25 March 2023).
60. Dusty-nv. GitHub. Available online: https://github.com/dusty-nv/jetson-inference (accessed on 25 March 2023).
61. Tan, R.J. Towards Data Science. 24 March 2019. Available online: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (accessed on 25 March 2023).
62. Subramanyam, V.S. Medium. 17 January 2021. Available online: https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef (accessed on 25 March 2023).
63. Tensorflow 2 Detection Model Zoo. Available online: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (accessed on 25 March 2023).
64. Tensorflow. Available online: https://www.tensorflow.org/install/source (accessed on 25 March 2023).
65. Tensorflow Object Detection API Tutorial. Available online: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#tf-install (accessed on 25 March 2023).
66. Bhande, A. Medium. 11 March 2018. Available online: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76 (accessed on 25 March 2023).
67. Mikołajczyk, A.; Grochowski, M. Data augmentation for improving deep learning in image classification problem. In Proceedings of the International Interdisciplinary PhD Workshop (IIPhDW), Swinoujscie, Poland, 9–12 May 2018; pp. 117–122. [CrossRef]
68. Github. Available online: https://github.com/NVIDIA/TensorRT (accessed on 25 March 2023).

69. Neta, Z.; Wu, H.; Rodge, J. NVIDIA. 20 July 2021. Available online: https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/ (accessed on 25 March 2023).