Journal of
Low Power Electronics and Applications

Article

Real-Time Embedded Implementation of Improved Object Detector for Resource-Constrained Devices

Niranjan Ravi and Mohamed El-Sharkawy *

Department of Electrical and Computer Engineering, Purdue School of Engineering and Technology, Indianapolis, IN 46202, USA; ravin@iu.edu
* Correspondence: melshark@purdue.edu

Abstract: Artificial intelligence (A.I.) has revolutionised a wide range of human activities, including the accelerated development of autonomous vehicles. Self-navigating delivery robots are recent trends in A.I. applications such as multitarget object detection, image classification, and segmentation to tackle sociotechnical challenges, including the development of autonomous driving vehicles, surveillance systems, intelligent transportation, and smart traffic monitoring systems. In recent years, object detection and its deployment on embedded edge devices have seen a rise in interest compared to other perception tasks. Embedded edge devices have limited computing power, which impedes the deployment of efficient detection algorithms in resource-constrained environments. To improve on-board computational latency, edge devices often sacrifice performance, creating the need for highly efficient A.I. models. This research examines existing loss metrics and their weaknesses, and proposes an improved loss metric that can address the bounding box regression problem. Enhanced metrics were implemented in an ultraefficient YOLOv5 network and tested on the targeted datasets. The latest version of the PyTorch framework was incorporated in model development. The model was further deployed using the ROS 2 framework running on NVIDIA Jetson Xavier NX, an embedded development platform, to conduct the experiment in real time.

Keywords: neural networks; YOLOv5; deep learning; ROS 2; CNN; object detection; NVIDIA; NVIDIA Jetson Xavier NX; ROS; PyTorch

check for updates

Citation: Ravi, N.; El-Sharkawy, M. Real-Time Embedded Implementation of Improved Object Detector for Resource-Constrained Devices. *J. Low Power Electron. Appl.* **2022**, *12*, 21. https://doi.org/10.3390/jlpea12020021

Academic Editors: Nicu Bizon and Mihai Oproescu

Received: 15 February 2022
Accepted: 31 March 2022
Published: 13 April 2022

1. Introduction

The automotive and transportation industry has changed our lives, culture, and climate more than any other innovation and technical development in human history has. Level 5 vehicle autonomy, sometimes also referred to as fully connected autonomous vehicles, is an advanced research topic in engineering science. A self-driving car can visualise and sense its surroundings with the help of cameras, lidar, and other sensor devices. In autonomous vehicles, data generated by the sensing devices are large [1] and demand the development of efficient algorithms to process information and reach intelligent decisions [2,3].

Computer-vision (CV) algorithms are utilised to provide a solution to such complex tasks. CV uses artificial neural networks (ANNs) or convolutional neural networks (CNNs) to address the problems of nonlinearity [4]. Applications of CV have been the most popular in the last decade and show a rising trend [5]. However, the deployment of state-of-the-art SSD [6], AlexNet [7], and RESNET50 [8] CNNs in resource-constrained edge devices is a complex problem [9]. Edge devices are limited in their computation capabilities [10] and are often battery-powered in remote locations [11]. Techniques such as model adaptive mechanisms for resource-constrained edge device, and hardware acceleration approaches for hardware and software were developed [12] to overcome the above challenge. The NVIDIA Jetson and Tegra families provide dedicated GPU and TPU to perform computation tasks in edge devices.

Object detection is one of the fundamental problems of computer vision that shapes the basis of other computer vision tasks, involving perception such as instance segmentation [13], image captioning [14], and object tracking. When performing noncritical computer-vision tasks requiring very high levels of accuracy, it is rational to trade off speed for increased network accuracy. Object detection comprises tasks of object classification and localisation or positioning. Two-stage object detectors depend on the presence or absence of regional proposal networks (RPNs). Single-stage object detectors such as SSD and YOLO [15] are popular choices of object detectors because of their high speed of inference, frames per second (FPS), and accuracy. SSD does not contain an RPN but uses a concept of prior predefined bounding boxes at different feature maps. One such object detection method is You Only Look Once (YOLO), which was introduced by R. Joseph et al. in 2015. It quickly gathered popularity as it was one of the leading single-stage detector in the deep-learning era that outperformed other state-of-the-art methods such as region-based convolutional neural networks (R-CNNs) and deformable part models (DPMs). The function of a YOLO model depends on a unified detection technique that fuses different modules into a single neural network. The YOLO architecture is widely used in various research applications ranging from the detection of smaller objects such as UAVs and face masks to multiobject autonomous detection [16].

We provide a brief background on object detectors and bounding box regression concepts in Section 2. Section 3 explores the network architecture of the object detector utilized in this research. Various existing loss metrics are presented and their drawbacks are studied in Section 4. A new loss metric is proposed in Section 4.4 to provide an optimal strategy for bounding box regression while overcoming the drawbacks of other metrics. A simulation experiment was carried out on synthetic data to observe the performance of the proposed loss metric in Section 5, followed by testing the YOLOv5 model in object detection datasets PASCAL VOC 2007 [17] and CGMU [18] in Section 9. The research was further extended to analyse the performance of the improved model in real-time situations with the use of the Jetson NX module and robotic operating system (ROS) 2. The hardware and software concepts are detailed in Section 6.

2. Background and Literature Review

2.1. Object Detectors

Object detectors use backbone architecture such as VGG16, VGG19 [19], and GoogLeNet [20] to extract crucial features from the image. Backbone architectures are trained in the ImageNet dataset consisting of about 1000 classes [21]. As network weights of the backbone layers improve the model's performance, they also increase the number of trainable parameters associated with the model. This creates a heavy CNN model that requires extensive computation resources [22]. In order to solve the drawbacks of existing object detectors, various small and lightweight object detectors such as MobileNetV2-SSDLite [23], YOLOv3 [24], FireNet [25], Light-Net [26], and Tiny-DSOD [27] were investigated. The size of the model and parameters associated with the neural network are reduced by compression techniques such as pruning [28] and quantisation [29], generating a lightweight model for resource-constrained platforms. Floating-point operations per second (FLOPS) indicate the number of operations performed by the GPU/CPU for the particular neural-network architecture. Owing to its complexity, object detection and segmentation frameworks contain a greater number of FLOPS and network parameters compared to that in classification tasks. It is necessary to establish a proper trade-off among accuracy, FLOPS, and network parameters to develop a small efficient model with better performance [27].

2.2. Bounding Box Regression Loss

Current object detection methods can be categorised into two types on the basis of the presence or absence of anchor boxes. Anchor-based detectors contain predefined anchors with prior scales and aspect ratios [30]. In contrast, anchor-free detectors utilise locations of corners and centroids to group into bounding boxes if they are geometrically aligned [31]. Bounding box parameters must be estimated to determine localisation boxes for objects in an image. Prior research works [32] adopted l_n normalisation losses for bounding box regression and are sensitive to bounding boxes of varying scales. Selective search algorithms are used to predict bounding box coordinates by prediction location, and size offsets of prior bounding boxes [33]. This loss function was later replaced by IoU loss and its variants. Various studies focusing on addressing drawbacks of existing loss functions and providing alternative approaches were recently studied [31,34]. These developments have significantly improved the deployment of object detectors.

2.3. Performance Metrics

To evaluate object detector performance across different datasets, average precision (AP) and mAP are the most widely used metrics. mAP computes the difference between the ground truth and predictions of the network [35]. This section outlines background knowledge for mAP estimation with the use of precision and recall values. In order to compute precision and recall, the following concepts are utilised.

- True positive (TP): correct prediction matching ground truth coordinates.
- False positive (FP): incorrect or misplaced detection of an object.
- False Negative (FN): undetected ground truth coordinates.
- True Negative (TN): prediction when no ground truth exists.

Since an infinite number of bounding boxes without any object are predicted by the network inside a given image [36], *TN* predictions can be ignored in object detection. To establish the difference between correct and incorrect predictions, the IoU threshold is used. The IoU region calculates the overlap between prediction and ground truth. If the prediction result of the network IoU is $\geq$t, it is classified as a correct prediction. Predictions under the threshold limit are classified as incorrect.

The assessment of object detection is carried out with the use of precision (P) and recall (R) estimation.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Precision estimates how accurate the predictions of the network are, and recall estimates how well the model estimates the positives among ground truth variables. The precision–recall curve is a trade-off between precision and recall values for different confidence thresholds ranging from 0.5 to 0.95. An ideal object detector achieves high precision and high recall, indicating that it could identify all ground truth labels and relevant objects [37].

However, in practical cases, it is difficult to always achieve a higher area under the curve (AUC), indicating very high performance. The 11-point and all-point interpolation are performed to remove noise from the precision–recall curve, and *AP* is calculated for all classes in the dataset [32]. *mAP* is the average *AP* of all the classes at a certain overlap threshold.

$$mAP = \frac{1}{N}\sum_{i}^{N} AP_i \tag{3}$$

N represents total number of object classes, and *AP_i* indicates the average precision for each class.

3. YOLO

YOLO is a state-of-the-art single-stage object detector module that is lightweight and suitable for deployment in edge devices. YOLOv5 is the latest in this series, and it is applied here due to its accuracy, speed, and capability to detect objects in the first run. Identical to the task of object detection in image-based visual analysis, multiobject tracking is a vital component for examining videos. The Pytorch implementation of YOLOv5 belongs to the family of object detection architectures and models pretrained on the COCO dataset that are passed to a deep-sort algorithm to track objects. This can track any object that the YOLOv5 model was trained to detect. In this study, a video feed from cameras is processed at the edge device using the YOLOv5 model for multiobject detection [38].

The YOLOv5 model was released in 2020, and it contains variants such as YOLOv5n, YOLOv5s, YOLOv5m, and YOLOv5x. The YOLOv5 object detector module underwent multiple improvements from YOLOv1 to YOLOv4 to accelerate and enhance model performance. The baseline architecture is open-sourced and available at git URL [39]. YOLO network divides input images into N × N grids. Each grid is treated as a regression problem to determine the bounding boxes of objects in the image. This model significantly improved detection performance in the PASCAL and COCO [40] datasets. The architecture of the model is shown in Figure 1.

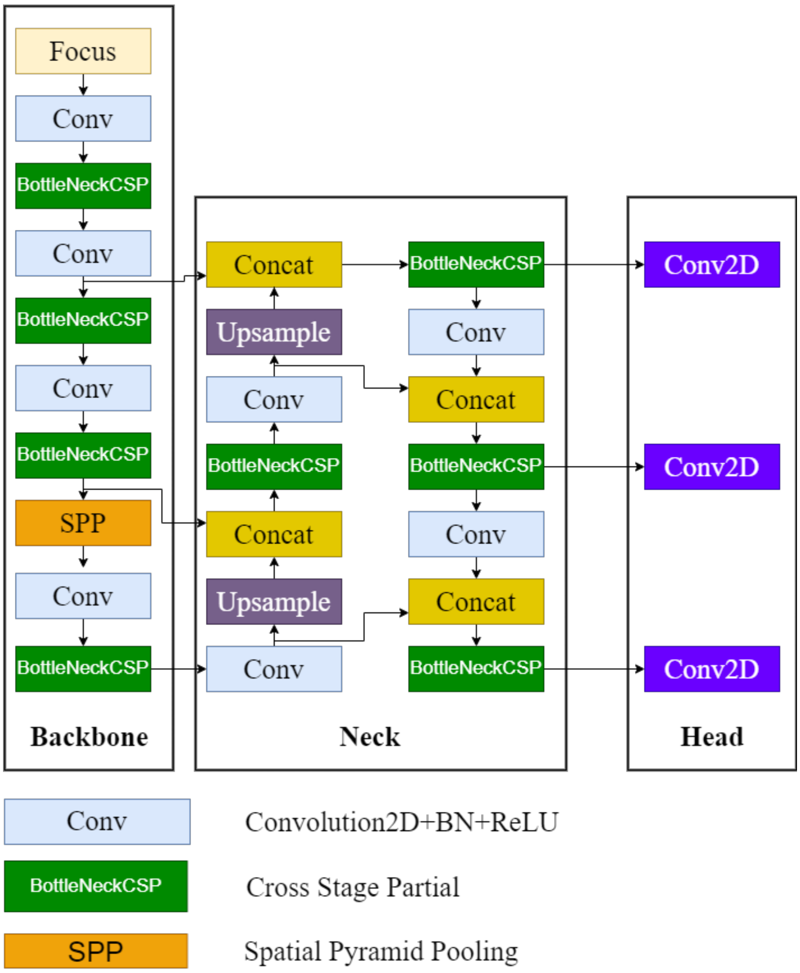


Figure 1. YOLOv5 network. Architecture [38].

There are three major components of the YOLOv5 network: backbone, neck, and head. In the backbone region of YOLOv5, a cross-stage partial network (CSPNet) is integrated into Darknet [41], creating CSPDarknet. CSPNet reduces the number of parameters and FLOPS of the model by solving problems associated with vanishing gradient. As shown in Figure 1, the backbone structure begins with a focus module to downsample input images. The backbone network extracts feature maps of various sizes from input samples through complex convolutional and pooling layers. Pooling layers reduce network complexity by reducing the dimensions of feature maps (downscaling) but retaining image characteristics.

The backbone utilises spatial pyramid pooling layers that execute pooling with different kernel sizes to effectively maintain an object's image characteristics and spatial information. Feature maps from the backbone are fused in the neck portion of the network. To increase information flow, YOLOv5 adopts PANet in the neck. PANet uses a feature pyramid structure (FPN) to fuse low- and high-level feature maps [42]. Localisation and robust semantic features are retained in the network with the use of FPN structures. The head region of the network obtains fused feature maps from the neck. As shown in Figure 1, three types of predictions are carried out in the head to attain multiscale predictions of small, medium, and large objects [43]. Nonmaximal suppression (NMS) is implemented on the head region to retain the best bounding box overlapping with the ground truth.

Both backbone and neck contain CSPDarknet layers that are different from each other. CSPDarknet layers in the backbone are equipped with residual units, while the latter contains convolutional layers. To increase the size of the feature maps, upsample block is utilised. The second is the prediction head, where the network can predict objects of varying shapes and sizes. This prediction technique is beneficial in scenarios of dense object detection such as traffic signs or a crowded walking zone. YOLOv5 offers different models, and YOLOv5n6 was chosen here because of its reduced size and better performance in benchmark object-detection datasets [31]. The model size of YOLOv5n6 is 6.6 MB, with 3.2 million parameters and 4.3 GFLOPS (1GFLOP = 10^9 $FLOPS$).

4. IoU Loss Metrics

Popular loss functions that are used as an evaluation metric for bounding box regression are studied in this section, followed by the proposal of an efficient loss function.

4.1. Intersection over Union (IoU)

In 2D/3D computer-vision challenges, IoU is a commonly used evaluation metric to find similarities or areas of intersection between two objects, namely, ground truth and prediction boxes represented as B^g and B^p [44]. Boxes consist of independent coordinates such as x_1, y_1, x_2, and y_2. They represent the edge coordinates of the boxes. These coordinates together represent the boundary limits of the object in the frame. As shown in Figure 2, x_1 , x_2, y_1, y_2 represent X_{min}, X_{max}, Y_{min} , Y_{max}, green represents the ground truth, and red the represents prediction box.

$$IoU = \frac{|B^p \cap B^g|}{|B^p \cup B^g|} \tag{4}$$

$$L_{IoU} = 1 - IoU \tag{5}$$

The numerator value in Equation (4) represents the intersection of ground truth and prediction boxes, and L_{IoU} represents the loss value. IoU loss is insensitive to the object's scale, symmetry, indiscernible identity, and the triangle inequality and experiences following drawbacks.

- Figure 3a represents cases of no overlap between predicted and ground truth boxes. In this scenario, the IoU metric performed poorly because $B^p \cap B^g \rightarrow 0$ and $IoU \rightarrow 0$, the gradient also became 0, and $L_{IoU} \rightarrow 1$, a very high loss value. During these cases, network training is halted. This can be observed as a primary drawback.
- The secondary drawback was observed during partial and complete overlap of bounding boxes where the L_{IoU} range was (0, 1). In Figure 3b, the overlap region remained the same even for prediction boxes of different sizes; in Figure 3c, a similar case can be observed, where the prediction box could be larger or smaller in size in comparison to the ground truth box. The metric does not regress on the basis of the aspect ratios and size of the bounding boxes. This leads to inaccurate predictions and errors in classification during dense object detections.

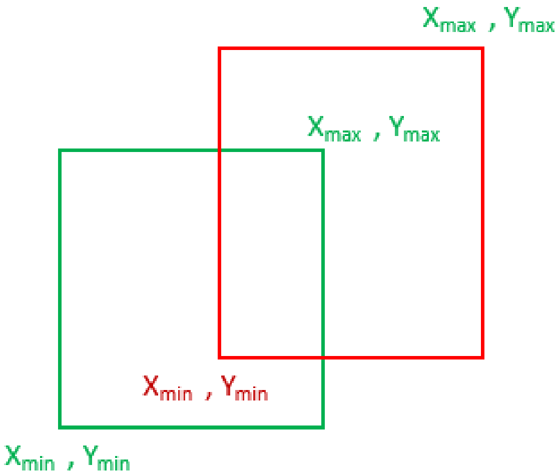


Figure 2. Bounding box coordinates.

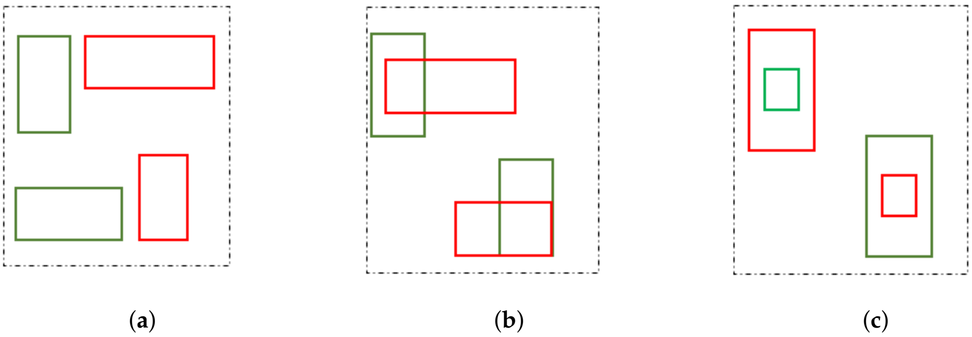


Figure 3. Representation of different scenarios of bounding box overlap. (**a**) No overlap; (**b**) partial overlap; (**c**) complete overlap.

4.2. Generalised Intersection over Union (GIoU)

To solve the existing drawbacks of *IoU*, an improved loss metric was proposed as follows.

$$L_{GIoU} = 1 - IoU + \frac{|C - (B^p \cup B^g)|}{|C|} \tag{6}$$

C is a minimum/small convex area enclosing both B and B^{gt}.

The primary drawback of *IoU* metric is loss values being high when there is no overlap between boxes. GIoU addresses this issue by calculating the small convex area that covers both ground truth and prediction boxes [45]. When there is no overlap, the gradient does not become 0, and the network would be able to regress in bringing the boxes closer together.

However, the *GIoU* metric had a downside during the complete overlap of the two boxes. As shown in Figure 3c, when the boxes overlapped, the minimal convex area equalled the union area. During this condition, *GIoU* loss degraded to *IoU* loss.

4.3. Distance Intersection over Union (DIoU)

To address the drawbacks faced by *IoU* and *GIoU* losses, and improve regression accuracy, *DIoU* is proposed [46].

$$L_{DIoU} = 1 - IoU + \frac{d^2(b, b^{gt})}{c^2} \tag{7}$$

$$c^2 = w^2 + h^2 \tag{8}$$

In Equations (7) and (8), b^{gt} and b represent centres of ground truth and prediction boxes, and c^2 is the diagonal length of the convex box enclosing both boxes. $DIoU$ loss takes the centre point distance into consideration while estimating the loss value. The distance estimation term acts as a penalty term in reducing distance between boxes and minimising regression loss.

Though $DIoU$ addresses drawbacks of previous losses, it fails to converge in the case of concentric rectangles where centres match, and one box is smaller than another. During these cases, $DIoU$ losses simply behave like IoU loss, leading to poor convergence and high regression errors [34].

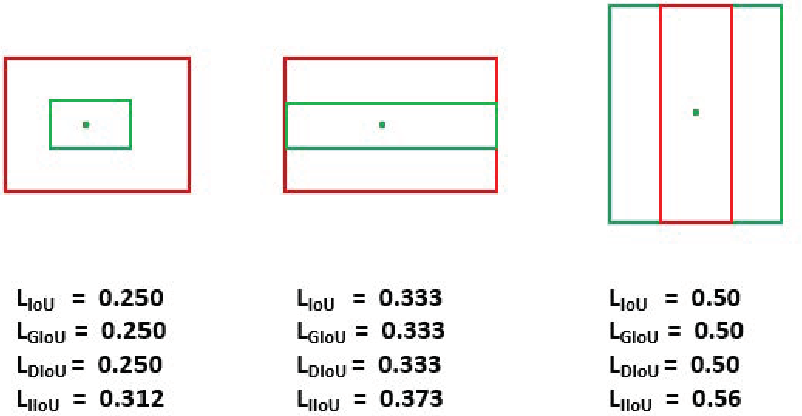


Figure 4. Three different cases when boxes have same centres but different aspect ratios and areas. Losses between boxes calculated by IoU, $GIoU$, $DIoU$ and $IIoU$ loss.

4.4. Improved IoU Loss (IIoU)

To addresses the drawback of $DIoU$ that occurs when centres of bounding boxes align at the same point, this research proposes a new loss function that estimates the distance between the central coordinates of boxes along the x and y axes.

$$L_{IIoU} = 1 - IoU + \frac{d^2(x^g, x^p)}{c^2} + \frac{d^2(y^g, y^p)}{c^2} \quad (9)$$

(x^g, y_1) and (x^p, y_1) represent the central coordinates along the x axis, and (y^g, x_1) and (y^p, x_1) indicate the centra; coordinates along the y axis.

Equation (9) can be explained by three parts: IoU loss, and distance along the axis lines of x and y centres. This improved metric calculates the distance between the central coordinates of the ground truth and prediction boxes. IIoU loss significantly differs from $DIoU$ since centres of the x and y axes are considered while calculating loss. In intersection cases with centrs aligned as shown in Figure 4, $GIoU$, $DIoU$ loss performs as an IoU loss, where the $IIoU$ metric reduces the difference between the aspect ratios of the boxes and helps the network in converging. Figure 4 represents the performance of all loss metrics in various scenarios where the centres of boxes aligned but aspect ratios were different. Algorithm 1 below represents sequential steps involved in estimating the loss metric. A simulation experiment was carried out to compare all loss metrics, detailed in Section 5.

Algorithm 1 Improved intersection over union loss estimation.

```
Input: $B^g = (x^g_1, y^g_1, x^g_2, y^g_2)$, $B^p = (x_1, y_1, x_2, y_2)$.
Output: $L_{IIoU}$.
Calculating union area of $B^g$ and $B^p$
$A^g = (x^g_2 - x^g_1) * (y^g_2 - y^g_1)$
$A^p = (x^p_2 - x^p_1) * (y^p_2 - y^p_1)$
Calculating intersection area between $B^g$ and $B^p$
$x^{max}_1$ = max($x^p_1$, $x^g_1$) , $x^{min}_2$ = min($x^p_2$, $x^g_2$)
$y^{max}_1$ = max($y^p_1$, $y^g_1$) , $y^{min}_2$ = min($y^p_2$, $y^g_2$)
if $x^{min}_2 > x^{max}_1$, $y^{min}_2 > y^{max}_1$ then
    I = ( $x^{min}_2 - x^{max}_1$ ) $*$ ($y^{min}_2 - y^{max}_1$ )
else
    I = 0
end if
Finding small enclosing convex box
$x^c_1$ = min($x^p_1$, $x^g_1$) , $x^c_2$ = max($x^p_2$, $x^g_2$)
$y^c_1$ = min($y^p_1$, $y^g_1$) , $y^c_2$ = max($y^p_2$, $y^g_2$)
$C^2_w = (x^c_2 - x^c_1$ ) ** 2 , $C^2_h = (y^c_2 - y^c_1$ ) ** 2 , $c^2 = C^2_w + C^2_h$
$A^c = (x^c_2$ - $x^c_1$ ) $*$ ( $y^c_2$ -$y^c_1$ )
$IoU$ = I/($A^g + A^p$ - I )
$GIoU = IoU - (A^c - (A^g + A^p - I))/A^c$
Calculating central coordinates of boxes $B^g$ and $B^p$
$x^g_{cc} = (x^g_2 + x^g_1)/2$ , $x^p_{cc} = (x^p_2 + x^p_1)/2$
$y^g_{cc} = (y^g_2 + y^g_1)/2$ , $y^p_{cc} = (y^p_2 + y^p_1)/2$
Calculating Euclidean distance between central coordinates
$\rho^2$ $(x^g, x^p) = (x^g_{cc} - x^p_{cc})$ ** 2 + ($y^g_1$ - $y^p_1$) ** 2
$\rho^2$ $(y^g, y^p) = (y^g_{cc} - y^p_{cc})$ ** 2 + ($x^g_1$ - $x^p_1$) ** 2
$IIoU = IoU - \rho^2 (x^g, x^p)/ c^2 - \rho^2 (y^g, y^p)/c^2$
return IIoU
```

5. Simulation Experiment

With the use of synthetic data, a simulation experiment was performed to visualise the performance of the $IIoU$ loss metric with other metrics. The experimental procedure was adapted from [46], and various parameters associated with a bounding boxes such as distance, aspect ratio and scale were considered during the experiment. The central coordinates (x,y) of the target or ground truth boxes were fixed at (10,10). Seven target unit boxes were created with different aspect ratios (1:4, 1:3, 1:2, 1:1, 2:1, 3:1, and 4:1) while maintaining the width-to-height ratio as 1. Anchor boxes were uniformly distributed at 210 data points. Within a fixed radius, anchor boxes were generated around the centre of (10,10). Then, 210 data points were uniformly chosen to place anchor boxes with 7 different scales and aspect ratios. This included the cases of overlap and nonoverlap with target boxes that had initially been created. Areas of anchor boxes were set to 0.5, 0.67, 0.75, 1, 1.33, 1.5, and 2 at different stages of iteration. Similar to the target box, anchor boxes were also provided with the same range of aspect ratios (1:4, 1:3, 1:2, 1:1, 2:1, 3:1, and 4:1). There were 115 anchor points, 7 aspect ratios, and 7 different scales adding to 5635 iterations. There were 7 different target boxes, and 5635 $*$ 7 = 39,445 was the total number of iterations for the simulation study.

In each iteration of Algorithm 2, the gradient descent algorithm was adapted for bounding box regression. Prediction at given iteration t is obtained by

$$B^t_{n,s} = B^{t-1}_{n,s} + \alpha(2 - IoU^{t-1}_{n,s})\nabla B^t_{n,s} \tag{10}$$

$B^t_{n,s}$ corresponds to the prediction box at iteration t, and $\nabla B^t_{n,s}$ denotes gradient loss at iteration $(t-1)$. $\alpha(2 - IoU^{t-1}_{n,s})$ is multiplied with gradient loss for faster convergence in all loss functions. The final regression error was evaluated using l_1-norm.

Algorithm 2 Simulation experiment on synthetic data.

Input: $\{\{B_{n,s}\}^S_{s=1}\}^N_{n=1}$ indicates anchor boxes at 115 scattered points within the central point at (10,10). S = 7 × 7 covering 7 different scales and aspect ratio of the anchor boxes. $\{B^{gt}_i\}^7_{i=1}$ is the set of ground truth boxes with centre (10,10) and 7 aspect ratios. α corresponds to learning rate.

Output: Regression error $E \in R^T$ is calculated for each iteration and 115 scattered points.

```
Initialise $E = 0$ and iteration limit of $39,445$ $(T)$ .
for $t = 1 \rightarrow T$ do
    for $n = 1 \rightarrow N$ do
        for $s = 1 \rightarrow 7$ do
            for $i = 1 \rightarrow 7$ do
                if $\alpha \leq 80\%T$ then $\alpha = 0.1$
                else if $\alpha \leq 90\%T$ then $\alpha = 0.01$
                else $\alpha = 0.001$
                end if
                $\nabla B^t_{n,s} = \partial L(B^{t-1}_{n,s}, B^{gt}_i) / \partial B^{t-1}_{n,s}$
                $B^t_{n,s} = B^{t-1}_{n,s} + \alpha(2 - IoU^{t-1}_{n,s})\, \nabla B^t_{n,s}$
                $E(t) = E(t) + |B^t_{n,s} - B^{gt}_i|$
            end for
        end for
    end for
end for
return E
```

Figure 5 shows the convergence speed of different loss functions over 39,000 iterations. THe IoU loss metric had the lowest convergence because the gradient remained 0 in cases where boxes did not overlap. The performance of $DIoU$ loss was better than that of $GIoU$ and concurrent with results of the literature [46]. The proposed $IIoU$ metric also converged at a slightly faster rate compared to the $DIoU$ metric. For instance, at iteration 40, IIoU had a loss value of $2.07 * 10^4$ when compared to $DIoU$ loss, which had $3.516 * 10^4$. $DIoU$ and $IIoU$ loss converged at iterations 161.8 and 162.6 with total cumulative errors of $5.38 * 10^6$ and $4.12 * 10^6$, respectively. This indicates that the proposed metric outperformed the previous metric with better performance and almost the same cost.

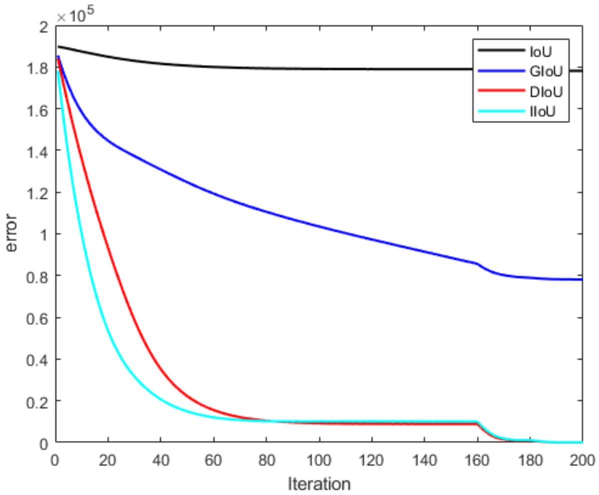


Figure 5. Simulation results of regression errors of different loss functions at iteration t.

We further extended the evaluation experiment of the proposed loss metric on object detection datasets explained in Section 7.

6. Embedded Deployment

This section details the technical configurations of ROS, embedded software, and Jetson NX, a low-powered embedded platform designed by NVIDIA for GPU computing in edge devices.

6.1. Robot Operating System(ROS)

ROS is an open-source software development kit for robotics applications. It provides services of a traditional operating system such as hardware abstraction, device control at a low level, cross-functionality between various systems, message passing between processes, and the management of data packets. The ROS communication infrastructure manages many individual operations. It also provides services for transmitting data packets via topics that are extremely helpful in computer vision when many sensors are involved [47].

The sections below briefly provide an introduction to three levels of essential concepts in understanding ROS.

6.1.1. ROS Filesystem Level

ROS packages consist of ROS run-time processes called nodes, an ROS-dependent library, and configuration files. Packages contain all essential information and data to a granular level for successfully building software. A significant functionality of the packages is providing the placeholder for ROS messages that are essential for transmitting and receiving information between individual nodes. As shown in Figure 6, each node can work independently and transmit messages when required.

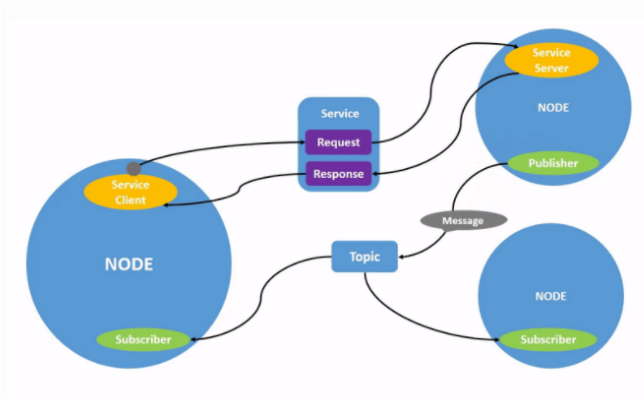


Figure 6. ROS nodes and messages.

6.1.2. ROS Computation

ROS nodes are single units, a modular structure designed to perform tasks. The processor acts as a primary or master node in a robotic architecture that receives information from individual sensor nodes. Messages are published in the ROS pipeline through topics. A node can subscribe to an issue and simultaneously publish data values to a different topic. These topics are instantaneously available in the ROS queue.

6.1.3. ROS Community Level

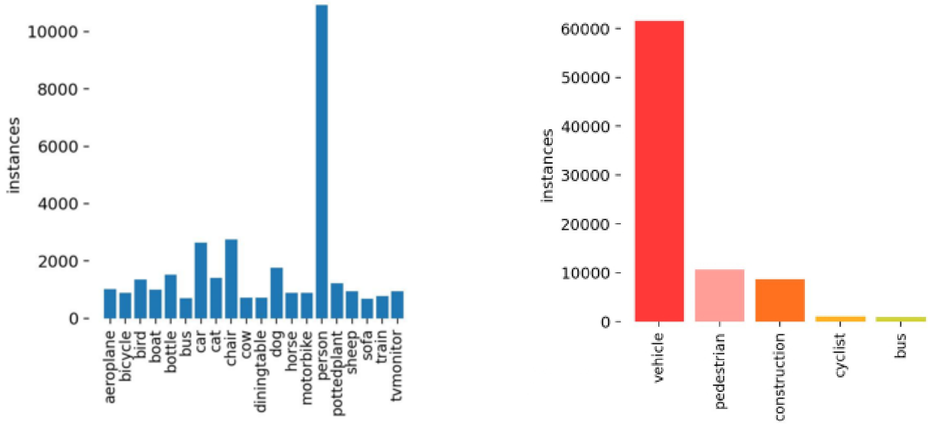
ROS community provides resources across various distributions of the framework. It is easier to collect the software and cross-compare with different versions. Since ROS is open-source, it provides guidance and support to developers to release their custom packages for other applications.

The latest version of ROS framework ROS 2 was utilised in this research. To gain advantages of packages from earlier revisions, the ros1-bridge module acts as helper node. The developed neural network can be used for various applications such as a traffic monitoring systems from a control room or placing an edge device in a remote location to observe traffic flow. Real-time model performance and evaluation with network weights of the architecture were tested in NVIDIA devices. The following section details the complete description of the hardware and software setup required for deployment.

6.2. Embedded System

Jetson Xavier NX was used because of its low cost and capacity for handling graphical and computational tasks. This device acts as a small artificial-intelligence (AI) computer. The size of the module is 70 × 45 mm. This module achieves very high performance while running on 10 watts of power. It is equipped with 40 expansion headers for external peripherals such as I2C and SPI. This device can deliver up to 21 tera operations per second (TOPS), so it is reliable for edge devices and systems. It is also equipped with 6 Carmel ARM CPUs, 48 tensor cores, 384 NVIDIA CUDA cores, and two NVIDIA deep-learning accelerator (NVDLA) engines. It can simultaneously process multiple neural-network nodes in parallel and high-resolution data from various sensors [48].

Power usage is reduced for sensors and peripheral devices while still enabling more software stacks on the device. The Jetson developer image was preconfigured with Ubuntu Linux ×64, version 18.04, so it is more feasible to install Pytorch and Tensorflow packages along with the needed Python libraries. The developer kit comprised 4 USB slots that were beneficial in our research.



(**a**) (**b**)

Figure 7. Distribution of objects in training dataset. (**a**) PASCAL VOC; (**b**) CGMU.

7. Dataset Preparation

Dataset preparation and annotation are very important factors in developing an efficient object detector. Two phases of training and evaluation were carried out with the use of the PASCAL VOC [17] and CGMU dataset [18]. In the first phase, PASCAL VOC 2007 and 2012 training and validation datasets were utilised for training the network, and PASCAL 2007 was used for testing the network. The training and validation dataset consisted of 16,551 images, and the testing dataset comprised 4952 images. The size of the images in the PASCAL dataset was 512 × 512. In the second phase, the CGMU dataset, acquired by road-side cameras at the city of Montreal, Canada, was utilised. The training and validation dataset consisted of 8007 images, and the testing dataset consisted of 1000 images. Since the CGMU dataset is very dense, image size was chosen to be 300 × 300. Figure 7 shows the distribution of objects in the PASCAL and CGMU datasets. Objects in the CGMU dataset are densely packed, thus hindering detection tasks.

PASCAL VOC dataset was separately trained with L_{IoU}, L_{GIoU}, L_{DIoU} and L_{IIoU} metrics on the YOLOv5n6 model. CGMU was similarly trained on YOLOv5s model. Performance evaluation is explained in Section 9.

8. Network Training and Testing Infrastructure

This section briefly details the setup for training and testing the YOLOv5 network.

8.1. Training Infrastructure

Network training was performed in Indiana University's large memory computer cluster called Carbonate. Carbonate provides specialised deep learning and GPU partitions to accelerate growth in deep-learning applications, and to provide access to students and research labs. Carbonate servers are managed by UITS Research Technologies to facilitate research activities [49]. The configuration of the training computer is listed below:

- Intel Xeon Gold 6126 12-core CPUs;
- 12 GPU-accelerated Lenovo ThinkSystem SD530 deep learning;
- Tesla V100;
- Operating system: UBUNTU 18.04;
- Pytorch Version 1.7.1;
- Python 3.7.3.

8.2. Testing Infrastructure

The Internet of Things (IoT) Collaboratory at Indiana University Purdue University at Indianapolis provided the infrastructure to evaluate this research. The evaluation portion of this research was conducted in two phases. In the first phase, the testing dataset was experimentally evaluated out to observe the performance improvement of the model. In the second phase, outdoor testing was carried out.

- NVIDIA Jetson NX;
- ROS2 Foxy version;
- Logitech USB camera;
- Tesla V100;
- NVIDIA GeForce GTX 1080;
- Linux4tegra operating system (Ubuntu-derived OS);
- Pytorch Version 1.7.1;
- Python 3.7.3.

9. Performance Evaluation

9.1. Performance Evaluation on PASCAL and CGMU Datasets

Network weights of the trained models with different loss metric were individually evaluated in this stage. The main performance measure utilised in this research was AP. AP = (AP50 + AP55 + AP60 + ... + AP95)/10, where each AP value indicates different IoU thresholds, namely, 50%, 55%, 60% , ..., 95%. All AP values are reported for loss metrics and are analysed. The confidence threshold for the evaluating networks was 0.001. Relative improvements were estimated for each loss metric on the basis of L_{IoU}. The final rows of Tables 1 and 2 indicate the relative improvement of L_{IIoU} in percentages.

Table 1 shows that L_{IIoU} displayed significant improvement in performance over the metrics. The relative improvement of the metric showed improved results across all thresholds, and the network demonstrated higher performance in AP50, AP60, and AP75, with an overall increase of 0.94% in AP. We could also observe that the AP50 of L_{IIoU} was at 78.6 and greater compared to that in the SSD512 architecture [6] trained in the PASCAL VOC dataset. Table 2 shows performance improvement in AP50, AP55, AP60, AP65, AP70, and AP.

9.2. Real-Time Testing and Analysis

Model weights from YOLOv5n6 (baseline)-improved YOLOv5n6 and SSD networks, which were trained and evaluated with PASCAL VOC dataset, are utilised in this section. Model weights were tested in real time using NVIDIA Jetson NX and ROS 2 middleware. ROS 2 Foxy was installed in Jetson, followed by a configuration of the ROS environment and setup of the workspace. ROS needed transport libraries to create a publisher camera node to access the USB camera. This node also creates a topic, USB/cameratopic, to publish the captured frames to the ROS queue. A subscriber node was designed for storing the incoming data from USB/cameratopic, and network weights of the architecture were transferred to the Jetson.

Table 1. Quantitative analysis of YOLOv5n6 trained using L_{IoU}, L_{GIoU}, L_{DIoU} and L_{IIoU}. Results reported on PASCAL VOC 2007 test dataset.

Loss Evaluation	**AP50**	**AP55**	**AP60**	**AP65**	**AP70**	**AP75**	**AP80**	**AP85**	**AP90**	**AP95**	**AP**
L_{IoU}	78.6	76.1	73.2	69.2	64	57.5	48.5	36.4	20.1	2.8	52.7
L_{GIoU}	78.5	76.3	73.2	68.6	63.7	57	48.9	36.4	21.1	3.4	52.7
Relative improvement	−0.1	0.2	0	0.4	−0.3	0.5	0.4	0	1	1.4	0
L_{DIoU}	78.6	76.3	73.2	69.1	63.6	57.3	49.5	37.4	21.6	3.3	53
Relative improvement	0	0.2	0	−0.1	−0.4	−0.2	1	1	1.5	0.5	0.3
L_{IIoU}	78.7	76.5	73.5	69.3	64.1	57.8	50.2	37.5	21.4	3.2	53.2
Relative improvement	0.1	0.4	0.3	0.1	0.1	0.3	0.7	1.1	1.3	0.4	0.5
Relative improvement%	0.13	**0.52**	**0.41**	0.14	0.15	**0.52**	0.15	**3.02**	**6.4**	**14**	**0.94**

Table 2. Quantitative analysis of YOLOv5s trained using L_{IoU}, L_{GIoU}, L_{DIoU} and L_{IIoU}. Results reported on CGMU test dataset.

Loss Evaluation	**AP50**	**AP55**	AP60	**AP65**	**AP70**	**AP75**	**AP80**	**AP85**	**AP90**	**AP95**	**AP**
L_{IoU}	48.8	45.8	42.4	38.3	32.3	27.7	20.7	13.6	6.2	0.79	27.7
L_{GIoU}	49.6	46	42.9	38.6	34.2	28.5	22.1	14.7	5.6	1.0	28.3
Relative improvement	0.8	0.2	0.5	0.3	1.9	0.8	1.4	1.1	−0.6	0.21	0.6
L_{DIoU}	49.5	46.5	43.1	39	33.2	28.3	22.2	13.2	6.4	1.2	28.3
Relative improvement	0.7	0.7	0.7	0.7	0.9	0.6	1.5	−0.4	0.2	0.41	0.6
L_{IIoU}	52.1	48.6	44.6	40.5	35.5	28	18.5	11.6	4.6	0.6	28.5
Relative improvement	3.3	2.8	2.2	2.2	3.2	0.3	−2.2	−2	−1.6	−0.19	0.8
Relative improvement%	**6.76**	**11.6**	**5.18**	**5.74**	**9.9**	0.11	−10.6	−14.7	−25.8	−24	**2.89**

A USB camera was installed on the Jetson module to test the model in real time. When the ROS shell script had been activated, the camera started publishing frames to the topic. The work flow of the detection module is shown in Figure 8. The subscriber node that captured the frame downsampled it to 512 × 512 before processing it. Detection results were stored in a local repository. This process was repeated for YOLOv5n6 (baseline), improving the YOLOv5n6 and SSD architectures. Experiments were repeated in multiple outdoor environments ranging from busy crowded areas to empty parking lots, and FPS was observed. Figure 9 represents the average FPS that we obtained for each architecture, and model parameters in millions. The YOLOv5n6 model operated at approximately 25 FPS as compared to SSD operating at 7 FPS. A higher number of parameters associated with SSD is a factor to low FPS. The object detection results and corresponding bounding boxes of SSD, and improved YOLOv5n6 models are shown in Figure 10. The SSD network was unable to detect more objects and produced inaccurate predictions Figure 10a as compared to YOLOv5n6, shown in Figure 10b.

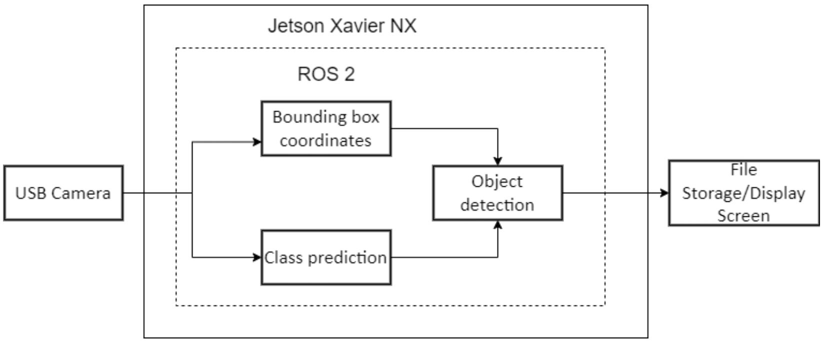


Figure 8. System workflow for real-time object detection.

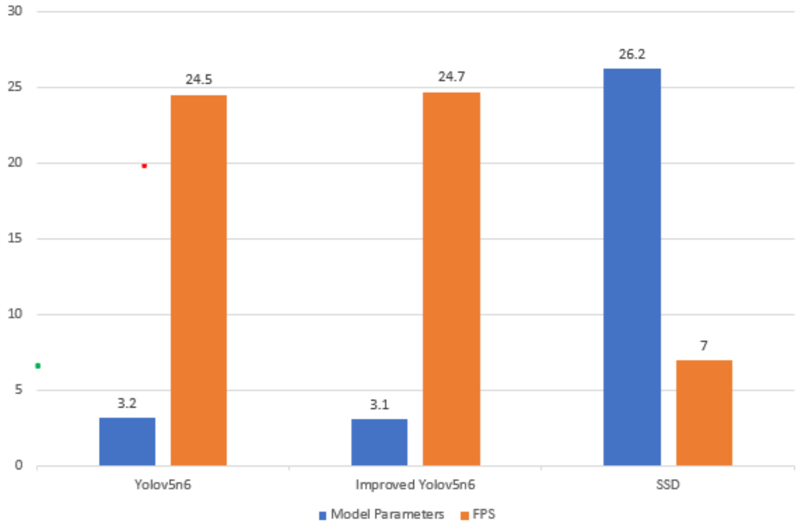
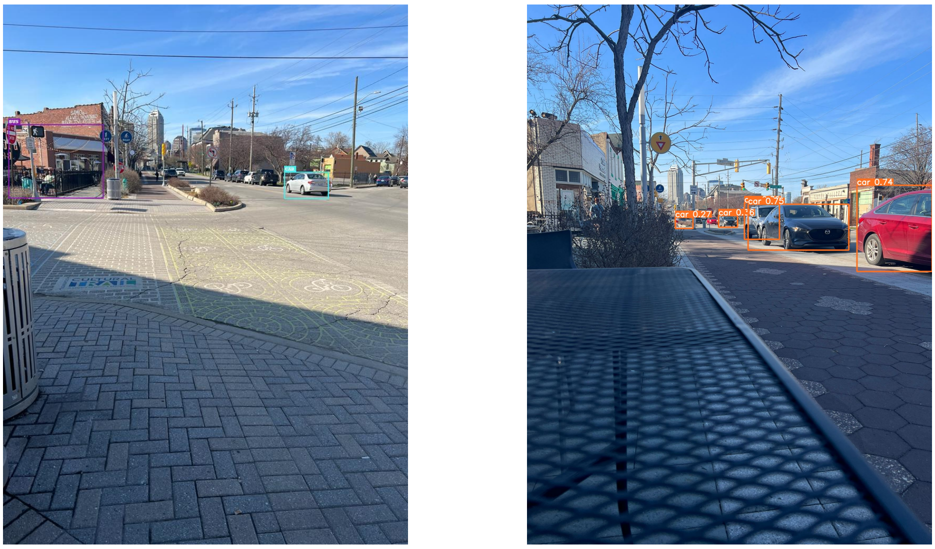


Figure 9. Performance evaluation of YOLOv5n6, improved YOLOv5n6, SSD architectures in real-time outdoor environments.



(**a**) (**b**)

Figure 10. Comparison of detection results between SSD and Improved YOLOv5n6 in real time. (**a**) Real-time detection results from SSD. (**b**) Real-time detection results from improved YOLOv5n6.

10. Conclusions

In this research, recent trends and developments in object detection and bounding box regression were addressed along with challenges associated with deploying complex CV algorithms on resource-constrained devices. An improved loss metric was proposed after evaluating existing loss metrics. The proposed metric was implemented in a lightweight YOLOv5 network, and laboratory evaluation experiments performed on PASCAL 2007 and CGMU test datasets showed improved performance. Further experiments were carried out to analyse the efficiency of the research in real time. The network model was deployed in an embedded framework, NVIDIA Jetson NX, with ROS 2. Future works could involve developing a semiautonomous robot with the proposed metric, with additional sensors such as LIDAR and RADAR.

Author Contributions: Conceptualization, N.R. and M.E.-S.; methodology, N.R.; software, N.R.; validation, N.R. and M.E.-S.; writing—original draft preparation, N.R.; writing—review and editing, N.R. and M.E.-S.; supervision, M.E.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: We have utilized open-source datasets such as PASCAL VOC and CGMU. They are cited in references [17,18] of the article.

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

References

1. Li, C.; Luo, Q.; Mao, G.; Sheng, M.; Li, J. Vehicle-mounted base station for connected and autonomous vehicles: Opportunities and challenges. *IEEE Wirel. Commun.* **2019**, *26*, 30–36. [CrossRef]
2. Venkitachalam, S.; Manghat, S.K.; Gaikwad, A.S.; Ravi, N.; Bhamidi, S.B.S.; El-Sharkawy, M. Realtime Applications with RTMaps and Bluebox 2.0. In Proceedings of the on the International Conference on Artificial Intelligence (ICAI), Las Vegas, USA, 30 July–2 August 2018; pp. 137–140.
3. Chitanvis, R.; Ravi, N.; Zantye, T.; El-Sharkawy, M. Collision avoidance and Drone surveillance using Thread protocol in V2V and V2I communications. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019; pp. 406–411. [CrossRef]
4. Katare, D.; El-Sharkawy, M. Embedded System Enabled Vehicle Collision Detection: An ANN Classifier. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 0284–0289. [CrossRef]
5. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
6. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
7. Alex, K.; Sutskever, I.; Geoffrey, H. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90.
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
9. Toma, A.; Wenner, J.; Lenssen, J.E.; Chen, J.-J. Adaptive Quality Optimization of Computer Vision Tasks in Resource-Constrained Devices using Edge Computing. In Proceedings of the 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Larnaca, Cyprus, 14–17 May 2019.
10. Tsimpourlas, F.; Papadopoulos, L.; Bartsokas, A.; Soudris, D. A design space exploration framework for convolutional neural networks implemented on edge devices. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **2018**, *37*, 2212–2221. [CrossRef]
11. Ravi, N.; El-Sharkawy, M. Integration of UAVs with Real Time Operating Systems using UAVCAN. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 10–12 October 2019.
12. Borrego-Carazo, J.; Castells-Rufas, D.; Biempica, E.; Carrabina, J. Resource-Constrained Machine Learning for ADAS: A Systematic Review. *IEEE Access* **2020**, *8*, 4053–4059. [CrossRef]
13. Katare, D.; El-Sharkawy, M. Real-Time 3-D Segmentation on An Autonomous Embedded System: Using Point Cloud and Camera. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019.
14. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055.

15. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
16. Ieamsaard, J.; Charoensook, S.N.; Yammen, S. Deep Learning-based Face Mask Detection Using YoloV5. In Proceedings of the 2021 9th International Electrical Engineering Congress (iEECON), Pattaya, Thailand, 10–12 March 2021.
17. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
18. CGMU Dataset. Available online: https://donnees.montreal.ca/ville-de-montreal/images-annotees-cameras-circulation#additional-info (accessed on 14 February 2022).
19. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
20. Jahandad; Sam, S.M.; Kamardin, K.; Sjarif, N.N.A.; Mohamed, N. Offline Signature Verification using Deep Learning Convolutional Neural Network (CNN) Architectures GoogLeNet Inception-v1 and Inception-v3. *Procedia Comput. Sci.* **2019**, *161*, 475–483. [CrossRef]
21. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. Imagenet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
22. Zaidi, S.S.A.; Ansari, M.S.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A survey of modern deep learning based object detection models. *arXiv* **2021**, arXiv:2104.11892.
23. Zhao, Z.; Zhang, Z.; Xu, X.; Xu, Y.; Yan, H.; Zhang, L. A Lightweight Object Detection Network for Real-Time Detection of Driver Handheld Call on Embedded Devices. *Comput. Intell. Neurosci.* **2020**, *2020*, 6616584. [CrossRef] [PubMed]
24. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
25. Jadon, A.; Omama, M.; Varshney, A.; Ansari, M.S.; Sharma, R. FireNet: A specialized lightweight fire & smoke detection model for real-time IoT applications. *arXiv* **2019**, arXiv:1905.11922.
26. Rao, Y.; Yi, G.; Xue, J.; Pu, J.; Gou, J.; Wang, Q.; Wang, Q. Light-Net: Lightweight Object Detector. *IEEE Access* **2020**, *8*, 201700–201712. [CrossRef]
27. Li, Y.; Li, J.; Lin, W.; Li, J. Tiny-DSOD: Lightweight object detection for resource-restricted usages. *arXiv* **2018**, arXiv:1807.11013.
28. Reed, R. Pruning algorithms-a survey. *IEEE Trans. Neural Netw.* **1993**, *4*, 740–747. [CrossRef]
29. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
30. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1137–1149. [CrossRef]
31. He, J.; Erfani, S.; Ma, X.; Bailey, J.; Chi, Y.; Hua, X.S. α-IoU: A Family of Power Intersection over Union Losses for Bounding Box Regression. *Adv. Neural Inf. Process. Syst.* **2021**, *34*. Available online: https://papers.nips.cc/paper/2021/hash/a8f15eda80c50adb0e71943adc8015cf-Abstract.html (accessed on 14 February 2022).
32. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
33. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
34. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision,Venice, Italy, 22–29 October 2017.
35. Padilla, R.; Netto, S.L.; Da Silva, E.A. A survey on performance metrics for object-detection algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020.
36. Hanley, J.A.; McNeil, B.J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* **1982**, *143*, 29–36. [CrossRef]
37. Kolawole, S.; Osakuade, O.; Saxena, N.; Olorisade, B.K. Sign-to-Speech Model for Sign Language Understanding: A Case Study of Nigerian Sign Language. *arXiv* **2021**, arXiv:2111.00995.
38. Zhu, L.; Geng, X.; Li, Z.; Liu, C. Improving YOLOv5 with Attention Mechanism for Detecting Boulders from Planetary Images. *Remote Sens.* **2021**, *13*, 3776. [CrossRef]
39. Available online: https://github.com/ultralytics/yolov5 (accessed on 14 February 2022).
40. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014.
41. Zhang, D.; Fan, S.S. Fault detection method of transmission line based on Yolo V3. *Autom. Technol. Appl.* **2019**, *38*, 125–129.
42. Tang, J.; Liu, S.; Zheng, B.; Zhang, J.; Wang, B.; Yang, M. Smoking Behavior Detection Based On Improved YOLOv5s Algorithm. In Proceedings of the 2021 9th International Symposium on Next Generation Electronics (ISNE), Changsha, China, 9–11 July 2021.
43. Yang, G.; Feng, W.; Jin, J.; Lei, Q.; Li, X.; Gui, G.; Wang, W. Face mask recognition system with YOLOV5 based on image recognition. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020.
44. Yu, J.; Jiang, Y.; Wang, Z.; Cao, Z.; Huang, T. Unitbox: An advanced object detection network. In Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 15–19 October 2016.

45. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
46. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34.
47. Rivera, S.; Iannillo, A.K.; Lagraa, S.; Joly, C.; State, R. ROS-FM: Fast Monitoring for the Robotic Operating System (ROS). In Proceedings of the 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), Singapore, 28–31 October 2020.
48. Ciobanu, A.; Luca, M.; Barbu, T.; Drug, V.; Olteanu, A.; Vulpoi, R. Experimental Deep Learning Object Detection in Real-time Colonoscopies. In Proceedings of the 2021 International Conference on e-Health and Bioengineering (EHB), Iasi, Romania, 18–19 November 2021.
49. Stewart, C.A.; Welch, V.; Plale, B.; Fox, G.; Pierce, M.; Sterling, T.; Indiana University Pervasive Technology Institute. IUScholar-Works. Available online: https://scholarworks.iu.edu/dspace/handle/2022/21675 (accessed on 14 February 2022).