Article

On-Board Small-Scale Object Detection for Unmanned Aerial Vehicles (UAVs)

Zubair Saeed [1], Muhammad Haroon Yousaf [1,2,*], Rehan Ahmed [3], Sergio A. Velastin [4,5] and Serestina Viriri [6,*]

1 Swarm Robotics Lab (SRL), National Center of Robotics and Automation (NCRA), University of Engineering and Technology (UET), Taxila 47080, Pakistan; zubairsaeed602@gmail.com
2 Department of Computer Engineering, University of Engineering and Technology (UET), Taxila 47080, Pakistan
3 School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 24090, Pakistan
4 School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK
5 Department of Computer Science and Engineering, University Carlos III Madrid, 28911 Leganés, Spain
6 School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban 4041, South Africa
* Correspondence: haroon.yousaf@uettaxila.edu.pk (M.H.Y.); viriris@ukzn.ac.za (S.V.)

Abstract: Object detection is a critical task that becomes difficult when dealing with onboard detection using aerial images and computer vision technique. The main challenges with aerial images are small target sizes, low resolution, occlusion, attitude, and scale variations, which affect the performance of many object detectors. The accuracy of the detection and the efficiency of the inference are always trade-offs. We modified the architecture of CenterNet and used different CNN-based backbones of ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, Res2Net50, Res2Net101, DLA-34, and hourglass14. A comparison of the modified CenterNet with nine CNN-based backbones is conducted and validated using three challenging datasets, i.e., VisDrone, Stanford Drone dataset (SSD), and AU-AIR. We also implemented well-known off-the-shelf object detectors, i.e., YoloV1 to YoloV7, SSD-MobileNet-V2, and Faster RCNN. The proposed approach and state-of-the-art object detectors are optimized and then implemented on cross-edge platforms, i.e., NVIDIA Jetson Xavier, NVIDIA Jetson Nano, and Neuro Compute Stick 2 (NCS2). A detailed comparison of performance between edge platforms is provided. Our modified CenterNet combination with hourglass as a backbone achieved 91.62%, 75.61%, and 34.82% mAP using the validation sets of AU-AIR, SSD, and VisDrone datasets, respectively. An FPS of 40.02 was achieved using the ResNet18 backbone. We also compared our approach with the latest cutting-edge research and found promising results for both discrete GPU and edge platforms.

Keywords: object detection; computer vision; modified CenterNet; VisDrone; SSD; AU-AIR; object detectors; Jetson Xavier; Jetson Nano; NCS2

check for updates

Citation: Saeed, Z.; Yousaf, M.H.; Ahmed, R.; Velastin, S.A.; Viriri, S. On-Board Small-Scale Object Detection for Unmanned Aerial Vehicles (UAVs). *Drones* **2023**, *7*, 310. https://doi.org/10.3390/drones7050310

Academic Editors: Quanmin Zhu and Zain Anwar Ali

Received: 22 March 2023
Revised: 29 April 2023
Accepted: 30 April 2023
Published: 6 May 2023

1. Introduction

For many applications, such as surveillance and monitoring work, it is important to recognize objects in aerial images and to classify them. Benchmark datasets, such as Pascal VOC 2007, are very different because images from these datasets often only include one or a few objects that take up a large area of the image, whereas UAV-based images may contain a variety of objects that are small and distributed randomly. Detection from an aerial perspective is a challenging subject where researchers are still facing problems. When the drone reaches a specific height, it may obtain a wider field of vision and record more objects in one go. Viewing objects from a certain height affects their visibility as well as their size. For example, the size of a car is much clearer when we observe it physically, but if we observe it from a certain height, the size and visibility of it will change more as the viewpoint changes with height. Imagery captured by a UAV has larger scale attributes, and a traditional deep learning CNN-based object identification technique performs a size-reducing operation on the input image. It further reduces the information accessible for small objects and increases the complexity of detection [1]. Small-object recognition with high altitude and broad field of vision must be solved to enhance effectiveness.

For almost a decade, deep neural networks have dominated computer vision research and development on a variety of difficult tasks, i.e., semantic segmentation, object recognition with tracking, and image classification. Well-known state-of-the-art (SOTA) techniques such as RCNN, YOLO, and their derivatives have exhibited good accuracy on a variety of challenging datasets [2]. The scientific community has been more interested in aerial object detection as a result of rising interest and the generation of databases of drone-based images. However, the aforementioned methods that are SOTA object detection algorithms for general object detection datasets are less effective for more challenging situations such as those of aerial object datasets such as VisDrone, SSD, and AU-AIR.

Furthermore, processing resources are often limited, making existing time-consuming approaches that are beneficial for object detection, such as image pyramids, ineffective [3]. A coarse-to-fine pipeline is commonly employed with a coarse detector to identify large-scale instances and sub-areas that comprise densely distributed little ones, and a fine detector is subsequently applied to those areas to locate examples of small sizes. These algorithms offer promising results, but the coarse detector sub-regions can be rather rough, with a significant amount of background present, resulting in wasted calculations. Furthermore, because they divide the input image into many sub-images, they must process each sub-region individually, resulting in multiple inferences for each image. As a result, the disadvantages show that there is space for improved efficiency. To solve the aforementioned issues, we have provided an improved technique for object detection using three challenging datasets and implemented on cross-edge platforms after optimization.

The contributions of our research are presented later in detail, and they can be summarized as follows:

1. We altered the architecture of CenterNet and used nine different CNN-based backbones. Our modified CenterNet with hourglass104 as a backbone achieved the highest mAP, whereas ResNet18 obtained the highest FPS.
2. We implemented SOTA object detectors, i.e., YOLOV1 to YOLOV7, SSD-MobileNetV2, and Faster RCNN, and provided a comparison with the modified CenterNet.
3. We validated our methodology on three challenging datasets, namely VisDrone, SSD, and AU-AIR, and we were able to provide promising findings compared to the latest cutting-edge research and well-known object detectors.
4. We optimized object detectors and implemented them on edge platforms, i.e., Jetson Nano, Xavier, and NCS2. We found the best inference time using the modified CenterNet, and this research is an extended version of our previous work [4].

2. Related Work

One-stage and two-stage object detectors have been used in deep learning-based object detection models, and they have their own pros and cons. One-stage object detectors are efficient but usually have lower accuracy. Arunnehru L. et al. [5] proposed an improved YOLOv2 for accurate target detection in aerial vehicles by fine-tuning the network using a different self-collected dataset, changing the input size of the detector during each ten rounds of training and modifying the filter rules of the candidate box. These enhancements were created so that any aerial image of a UAV may be positioned in real time in the target region; the projection relationship can change the UAV's latitude and longitude automatically. YOLOv4 for aerial image identification was used in [6] that replaced the element-wise add operation with a concatenation operation in hybrid dilated convolution attention and solved the gridding problem using translational dilated convolution. The improved model was trained using the DOTA dataset, which contains objects ranging in size from 10 to 300 pixels. The approach performs well on DOTA, but it falls short of the present method in some scenarios for large-scale objects. Oghaz D. et al. [7] improved the efficiency of small object detection in their approach. The authors used feature augmentation modules such as super-resolution, deconvolution, and feature fusion. This was accomplished using SSD as the baseline network and smaller item detection on aerial data. These modules are intended to enhance the prediction layer's feature representation of small objects. Three datasets were used to evaluate the efficiency, including two aerial image datasets, with high accuracy but low FPS concentration that were mostly composed of small objects. A densely linked network was used by densely connected convolutional networks (DenseNet), where every layer was connected to another layer in a feed-forward method. An expansion layer and a squeeze layer made up the fire module from which SqueezeNet was built. The fire units were used as the core component of SqueezeNet, significantly reducing the number of parameters.

Several two-stage techniques for object detection and faster R-CNN rely on horizontal bounding boxes to precisely identify oriented items from UAV images. The final classification accuracy and localization accuracy become inconsistent as a result. There has been a lot of focus on this problem. For example, Xia et al. [8] created DOTA, which is a large-scale object detection benchmark dataset with oriented annotations as well. The majority of the available proposal-based object detection frameworks [9] form the foundation for many oriented object detectors. Setting rotated anchors [10], such as rotated RPN [11], in which the anchors with various angles, scales, and aspect ratios are placed on each position, is a logical alternative for oriented objects. Ding et al. presented the ROI transformer, which generates rotational ROIs from horizontal ROIs using RPN [9]. The detection accuracy of orientated objects is significantly increased in this way. However, fully linked layers and an ROI alignment procedure learning rotated ROIs resulted in the network becoming heavy and complex. Yang et al. [12] created an orientated object detection method based on the Faster RCNN, which is a general CNN-based object identification framework. Xu et al. [13] presented a label-oriented object format known as gliding vertexes, which acquired four vertex gliding offsets on the regression branch of the Faster R-CNN head. It has the ability to identify oriented objects.

Shams et al. [14] investigated the performance of single-board computer systems using CNN algorithms on the NVIDIA Jetson Nano, TX2, and R Pi. Suzen et al. [15] assessed the performance of the three frameworks, Caffe, TensorFlow, and apache, across a variety of hardware platforms to better understand the performance impact of the deep learning framework on various resources and to make recommendations for future hardware developments. Baller et al. [16] compared and analyzed how various deep learning frameworks perform on the Asus Tinker Edge R, R Pi, Coral Dev Board, and Jetson Nano in terms of inference time and power consumption. Verma et al. [17] introduced new ideas regarding optimization technology for the edge computing deep learning framework and proposed a thorough performance analysis across a range of hardware platforms. After optimization, the Jetson Nano-enhanced inference time for the MobileNetV2 neural network increased by 29.3%.

In the remaining part of this paper, we discuss the research methodology in Section 3, the experimental results with detailed discussion in Section 4, and the conclusion with possible future directions in Section 5.

3. Methodology

It is difficult to find small objects from an aerial perspective. Dealing with open-challenge datasets such as VisDrone, SSD, and AU-AIR makes it more difficult. Several cutting-edge detection models have problems performing well because of small object sizes, occlusion, unclear scenes, low resolution, and noisy background. Therefore, there is a need for a robust and accurate detection model that can accurately detect small size objects in an adequate time as well. In the proposed approach, we used three challenging datasets with default annotations. Then, we converted annotations according to the required format and split them into training, validation, and testing sets. Multiple deep-learning algorithms and a modified CenterNet with nine different CNN-based backbones for small-scale object detection were used to check their performance using discrete GPUs as shown in Figure 1. The trained models were optimized and implemented in different edge platforms, i.e., NCS2, Jetson Nano, and Jetson Xavier for onboard object detection. The bold arrows indicate the next phases of the overall methodology.

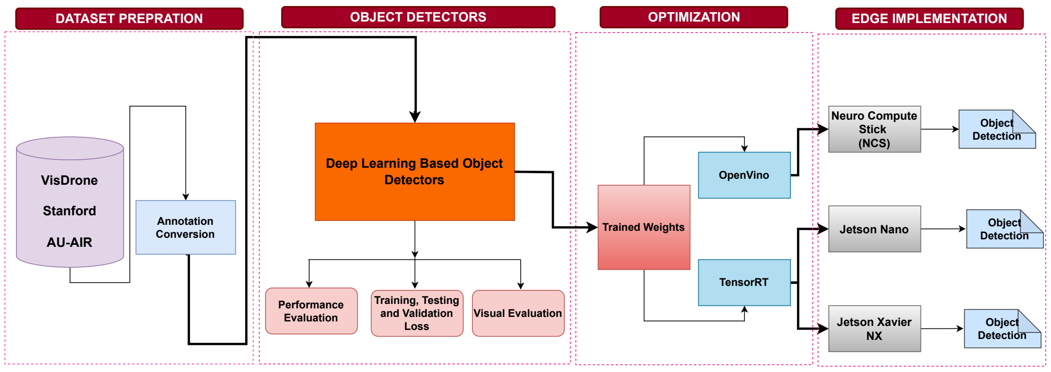

Figure 1. Block diagram as a broader view of our proposed approach.

3.1. Datasets

We used three challenging datasets, i.e., AU-AIR, SDD, and VisDrone, for the validation of modified CenterNet with different CNN-based backbones and SOTA object detectors. Figure 2 shows the complexity of three challenging datasets. We show car instances of cloudy weather of day, night, and the morning views of SDD, VisDrone, and AU-AIR datasets, respectively.

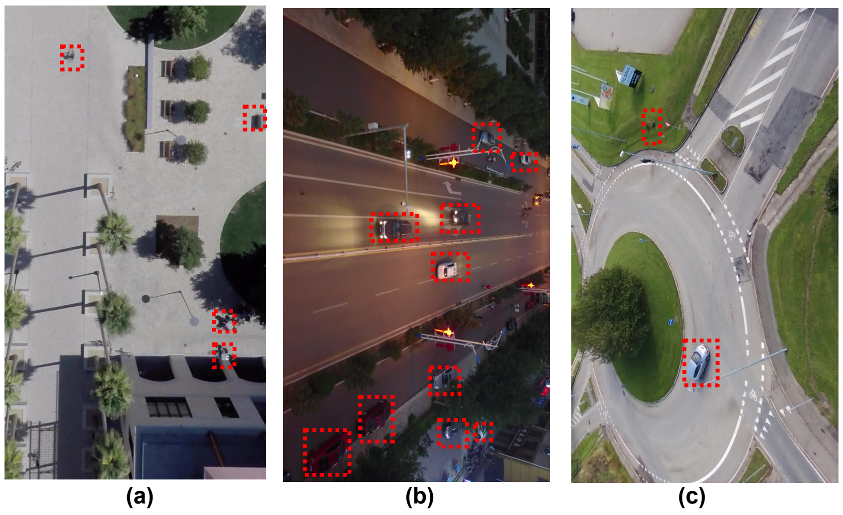

Figure 2. Samples of car instances from (**a**) Stanford Drone dataset, (**b**) VisDrone, and (**c**) AU-AIR datasets.

The VisDrone benchmark dataset [18] consists of 265,228 frames, which were captured by different UAV-based cameras and cover different types of situations. The dataset has a wide range of distances to objects, urban and rural environments, and eight different types of objects, such as pedestrian, person, car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle having dense, sparse, and crowded scenes. Bounding boxes for more than 2.6 million objects were manually annotated. This dataset was collected and annotated by a lab team named "machine learning and data mining" situated at a university in China. The Stanford Drone dataset (SDD) [19] is made up of over 100 height-view scenes with 20,000 targets taking part in different activities at the Stanford University campus. This benchmark dataset includes interactions between pedestrians, bikers, skateboarders, cars, and carts that followed social norms, with instances of around 11,200 pedestrians, 64 bicycles, 1300 hundred cars, 300 skateboarders, 2 carts, and 100 buses. AU-AIR was the first multi-modal dataset for UAV images [20] for small-scale objects. The dataset includes two hours of raw video captured from an aerial device having 32,823 labeled frames and eight objects for traffic surveillance purposes. The object instances include cars, vans, trucks, humans, trailers, bicycles, buses, and motorbikes. The dataset was initially used for traffic surveillance but can also be used for other purposes, i.e., tracking of automobiles, military, etc.

We prepared annotations of the datasets according to the required format. Annotations from the VisDrone dataset were distributed in the darknet (in.txt) format. Prior to converting them to COCO JSON, we changed them into Pascal VOC (in.xml) format. Annotations for the SDD dataset were also available in the darknet format, and we changed them into JSON format. Luckily, annotations for the AU-AIR dataset were already in COCO JSON and did not require conversion.

A modified CenterNet with nine different convolutional neural network-based backbones was used to extract important features. We trained them on the three datasets and then optimized them for Jetson Nano, Jetson Xavier NX, and Neuro Compute Stick (NCS2). We used tensorRT to optimize models for Jetson Nano and Xavier, while OpenVino was used to optimize models for NCS2. The broad view of CenterNet is shown in Figure 3. The CNN-based backbones acquire essential distinguished features and pass them to the modified CenterNet to detect objects from each dataset.

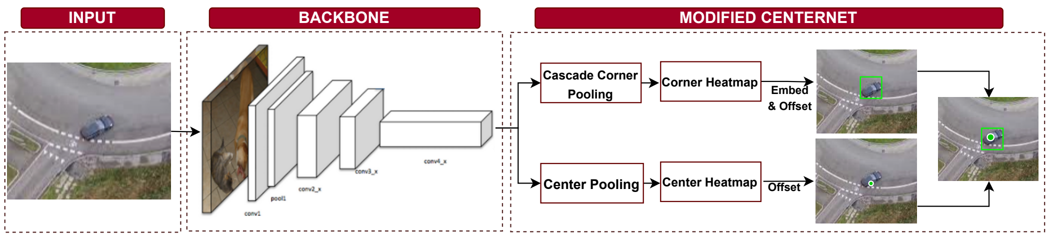

Figure 3. Modified CenterNet with convolutional neural network-based backbones.

3.2. CNN-Based Backbones

There were nine convolutional neural network-based backbones used with modified CenterNet. The generic architectures of residual neural network (ResNet), deep layer aggregation (DLA), and hourglass are illustrated in Figure 4.

A residual network (ResNet) [21] is an open-gated or gateless variation of the HighwayNet [22], the first exceedingly deep feedforward neural network that is practically feasible with hundreds of layers far deeper than preceding neural networks. By using a significant weight matrix for their gates, it can also be used to directly learn the skip weights. It is based on skip connections between layers. The majority of ResNet models are built using batch normalization between nonlinear double-layer or triple-layer skips (ReLU). ResNet's generic block diagram is displayed in Figure 4a. We used ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, Res2Net50, and Res2Net101 as the backbone with the modified CenterNet. We have seen that backbone with deeper layers achieved better accuracy during comparison.

A particular kind of convolutional encoder–decoder network is the hourglass network [23]. Convolutional layers are used to decompose and reconstruct inputs by deconstructing the image into a feature matrix. They take input in the form of images and extract features from them, as shown in Figure 4b. We used the last 104 layers of an hourglass to extract important features and pass them to the modified CenterNet.

The DLA (Deep Layer Aggregation) models are PyTorch implementations that are pre-trained on the ImageNet dataset. Their design uses repetition and grouping to reduce complexity. Blocks made up of layers are then divided into stages according to the feature resolution. The aggregation between the blocks and phases is the main priority, whereas the

DLA connects between stages, while HDA establishes connections both within and between stages. A generic DLA architecture is shown in Figure 4c with its essential components. We used DLA with 34 layers as a backbone with the CenterNet to extract features [24].

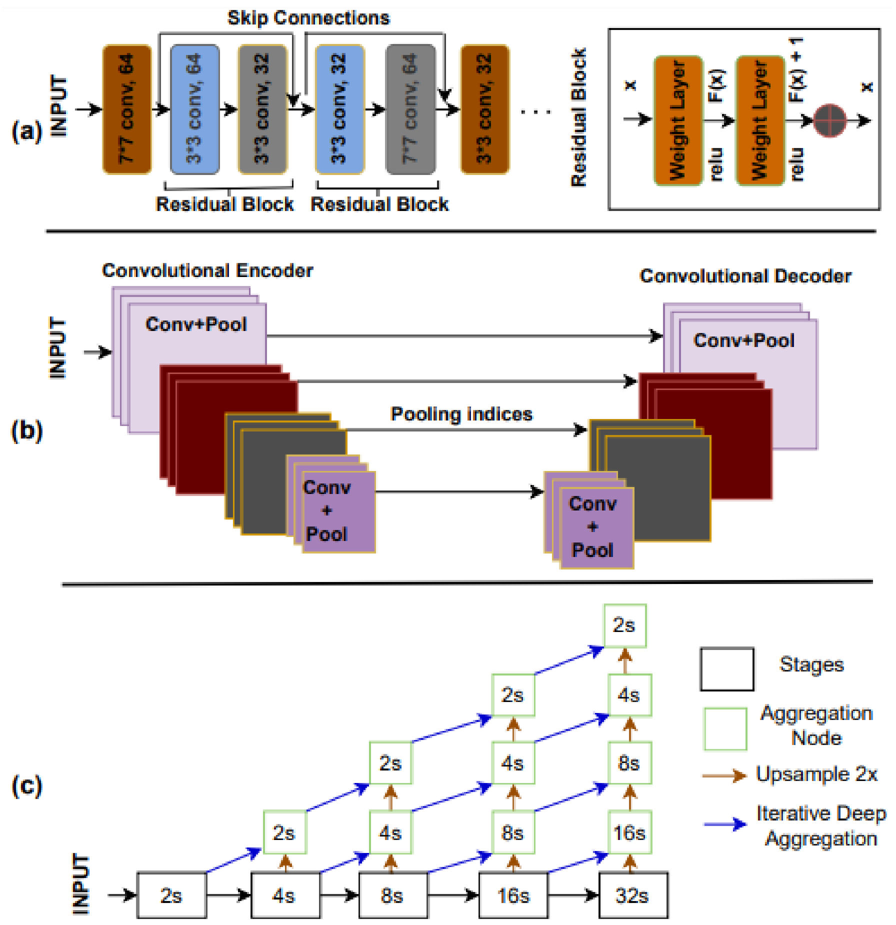

Figure 4. Generic architectures of convolutional neural network-based backbones used with a modified CenterNet: (**a**) ResNet; (**b**) Hourglass; (**c**) DLA.

3.3. Modified CenterNet

The traditional NMS (Non-Maximum Suppression) algorithm is replaced at the post-processing stage of the CenterNet [25] anchorless object detection architecture by a feasible method that works well with CNN flow, which is a significant advantage of the design. It is based on the concept that box predictions can be sorted based on the location of their centers instead of overlapping with the object. Each item is represented by a central keypoint and two corners. In particular, we incorporate a CornerNet-based heatmap for the center keypoints and estimate the offsets of the center keypoints. Instead of using a pair of key points to identify objects, our method uses a triplet. By doing this, it retains a one-stage detector while inheriting the ability of RoI pooling. The recommended approach takes into consideration the essential data at the lowest feasible cost. We implemented center pooling and cascade corner pooling to establish visual patterns inside the objects for keypoint detection. The size of the bounding box (bbox) core region contributes to the detection outcomes.

A comparatively large center area is produced by the scale-aware central region and vice versa for a large central portion. Let br_x and br_y represent $i's$ bottom-right corner, ul_x and ul_y represent $i's$ top-left corner, cul_x and cul_y represent $j's$ top-left corner, and cbr_x and cbr_y represent $j's$ bottom-right corner, as illustrated in Figure 5 where the other area represents the anticipated bounding boxes, and the yellow covered areas represent the scalable center regions. The relationship then needs to be matched by:

$$\begin{cases} \mathrm{cul_x} = \frac{(s+1)\mathrm{ul_x}+(s-1)\mathrm{br_x}}{2n} \\ \mathrm{cul_y} = \frac{(s+1)\mathrm{ul_y}+(s-1)\mathrm{br_y}}{2s} \\ \mathrm{cbr_x} = \frac{(s-1)\mathrm{ul_x}+(s+1)\mathrm{br_x}}{2s} \\ \mathrm{cbr_y} = \frac{(s-1)\mathrm{ul_y}+(s+1)\mathrm{br_y}}{2s} \end{cases} \quad (1)$$

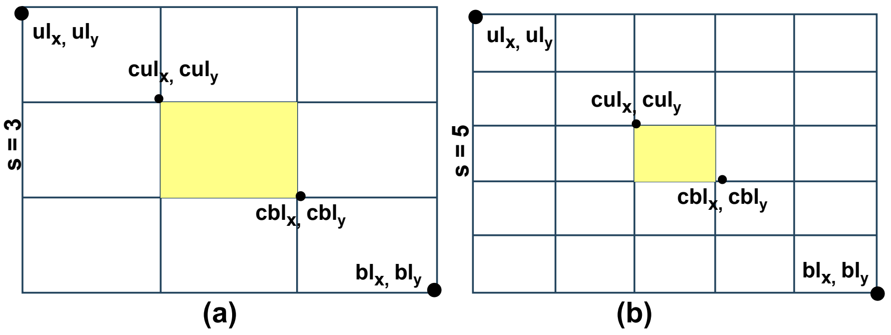

Figure 5. The central region when (**a**) s = 3 and (**b**) s = 5.

The odd number s, which is a central region, determines the scale. For the bounding box scales in our experimental research, we chose s values of 3 and 5. In order to identify the matching class instances, the center heatmap presents features estimation over the deep features from several downsampled CNN backbone frameworks. In order to determine the bbox center,

$$\widehat{\mathrm{O}}_{m,n,t} = \exp\left(-(m-\hat{p}_m)^2 + (n-\hat{p}_n)^2/2\sigma_p^2\right) \quad (2)$$

When $\widehat{O}_{m,n,t}$ has a value of 1, it indicates the center of a candidate key point; otherwise, it serves as the background, where the actual key points are m and n. $\hat{p}_m$ and $\hat{p}_n$ represent the predicted values of downsampled keypoints, where $\hat{p}$ stands for the object size-adaptive standard deviation, and t stands for the total number of classes.

The dimension head estimates the bbox's coordinates. The L1 norm can be used to estimate the size of the bbox for a class c and candidate object k with the coordinates $x_{min}, x_{max}, y_{min}, y_{max}$:

$$L1_{norm}(x,y) = (x_{max} - x_{min},\ y_{max} - y_{min}) \quad (3)$$

The offset and embedding offset are computed to lower the discretization error that arises due to downsampling the input. However, the center coordinates are calculated and applied to an input sample with higher dimensionality.

To enhance performance and to locate the impaired area with the related class, CenterNet proposes an end-to-end learning method that makes use of multi-task loss techniques. The proposed approach does this by applying a multi-task loss L to each sampled head:

$$L_{\text{C-Net}} = L_{\text{map}} + \lambda_{\text{dim}} L_{\text{dim}} + \lambda_{off} L_{off} \quad (4)$$

λ_{dim} and λ_{off} are constants with values of 0.1 and 1, respectively, whereas other parameters are as follows:

$$L_{\text{map}} = -\frac{1}{n}\sum_{i,j,c}\begin{cases} \left(1-\widehat{\mathrm{O}}_{i,j,c}\right)^{\alpha}\log\left(\widehat{\mathrm{O}}_{i,j,c}\right) \text{ if } \widehat{\mathrm{O}}_{i,j,c} = 1 \\ \left(1-\mathrm{O}_{i,j,c}\right)^{\beta}\left(\widehat{\mathrm{O}}_{i,j,c}\right)^{\alpha}\log\left(1-\widehat{\mathrm{O}}_{i,j,c}\right) \end{cases} \quad (5)$$

where n stands for the total keypoints. The center of the projected keypoint is represented by $\hat{\mathrm{O}}_{i,j,c}$, whereas the center of the actual candidate keypoint is determined by $\hat{\mathrm{O}}_{i,j,c}$. For each of our studies, the heatmap loss hyperparameters are α and β with values of 2 and 4, respectively.

$$L_{\text{dim}} = \frac{1}{n}\sum_{k=1}^{n}\left|\hat{b}_k - b_k\right| \tag{6}$$

$$L_{off} = \frac{1}{n}\sum_{p}\left|\widehat{F}_{\hat{p}} - (p/R^p - \hat{p})\right| \tag{7}$$

where p stands for the actual value, $\hat{p}$ stands for the key point that was downsampled during prediction, R stands for the output stride, and $\hat{F}$ stands for the projected offset value. In a similar way to this, b_k stands for the actual dimensions of bboxes obtained from ground facts, $\hat{p}_k$ for predicted bbox coordinates, and n for the total number of samples. The suspected images and the bbox are therefore input to the trained model. CenterNet uses this input to determine the class instance center points, offsets for the x and y coordinates, the dimension of the bboxes, and the related class.

3.4. *AI-on-the-Edge Implementation*

Table 1 shows the specification of the systems used during the experimentation. The discrete GPU has many cores, whereas NCS2 has only 16 cores. These cores and FLOPS ultimately determine the overall performance of detection models. The detailed comparison of results achieved using these GPU-based platforms is discussed in the next section as well. After evaluating the performance of the models on a discrete GPU, we evaluated inference time and performance using edge platforms, i.e., NCS, Jetson Nano, and Xavier.

Table 1. Specification of systems used during experimentation.

Parameters	**Jetson Xavier NX**	**Jetson Nano**	**NCS2**	**P5000**
Type	Embedded GPU	Embedded GPU	Embedded GPU	Discrete GPU
Architecture	Volta	Maxwell	Movidius	Pascal
Cuda Cores	384	128	16	2560
Memory	8 GB	4 GB	500 MB	16 GB
FLOPS	472 GFLOPS	472 GFLOPS	150 GFLOPS	8.9 TFLOPS
Power	10 W	5 W	2 W	180 W
Cost (USD)	400	100	90	2500

3.4.1. Neuro Compute Stick

Intel offers a toolbox called Open Visual Inference and Neural Network Optimization (OpenVino) to help deep learning models perform inference more quickly. It helps programmers in producing reliable and affordable computer vision applications. We installed all the dependencies required to optimize models in a separate virtual environment using a computer system without GPU. OpenVino allows for deep learning inference at the edge and supports diverse computer vision accelerator execution. We fed the pre-trained model into the model optimizer of OpenVino, and it transformed it into its .xml and .bin files as intermediate representations. The models are executed with the acceleration capabilities of the NCS2 using the inference engine of OpenVino as shown in Figure 6a. It controls the libraries needed for the code to work efficiently across many platforms. Since we used three datasets, we used the same resolution during optimization to keep the detection accuracy of the optimized model high, compromising detection speed (FPS).

3.4.2. Jetson Nano

TensorRT provides application programming interfaces (APIs) and parsers to import trained models from all primary deep learning frameworks and then creates efficient runtime engines that can be deployed in data centers as well as in automotive and embedded contexts. All the required packages and libraries were installed in a separate virtual environment. The trained models are fed into the tensortRT engine, which freezes the model and then optimizes and inferences the optimized model as shown in Figure 6b. We optimized the models for each dataset with the same resolution so that the detection accuracy would not be compromised.

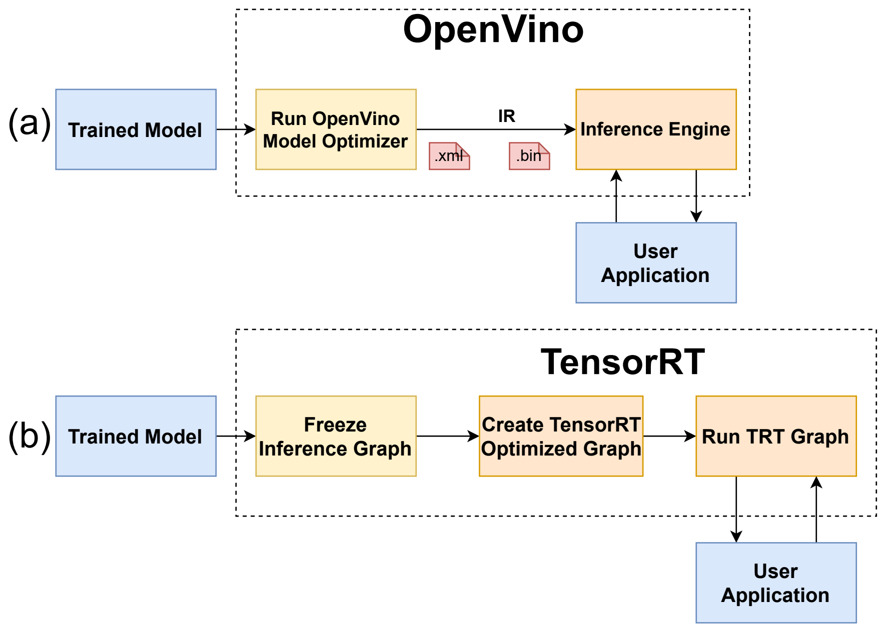

Figure 6. AI-Edge optimization and implementation: (**a**) OpenVino for NCS2 and (**b**) TensorRT for Jetson Nano and Xavier.

3.4.3. Jetson Xavier

We evaluated the performance of detectors at Jetson Xavier as well. The initialization procedure was the same as Nano and was followed according to Nvidia's official website. We created a separate virtual environment where all the required dependencies were installed. We optimized each model using TensorRT and evaluated their performance. A detailed comparison is shown in the next section.

4. Experimentation and Results Discussion

4.1. Experimental Setup

With an Intel(R) Core(TM) i5-9400, 16 GB RAM, and a CPU clocked at 2.90 GHz employing P5000 GPU capabilities, we trained and validated our approach. The Anaconda virtual environment was created where all the required libraries and packages were installed, i.e., PyTorch, Caffe, Darknet, Python3.8, Cuda, TensorRT, Pycocotools for windows, NumPy, Numba, etc. The modified CenterNet, YoloV5, YoloV6, YoloV7, and Faster RCNN were implemented using PyTorch. YoloV1 to YoloV4 were implemented using Darknet and SSD-MobilenetV2 using the Caffe framework. The batch size during training was two images for VisDrone, four images for AU-AIR, and one image for the SSD dataset against every deep learning model. We kept the default parameters of state-of-the-art object detectors, i.e., Yolo family, SSD-MobileNetV2, and Mask-RCNN, and the input size of the datasets during training and validation was kept according to the original size of the images. The modified CenterNet was trained and validated using the following parameters: 100 epochs, a stochastic gradient descent (SGD) for convergence, a learning rate of 0.15625×10^{-4}, a minimum IOU of 0.3, a momentum of 0.95, and a weight decay of 0.0005.

The local system was used as a host device with NCS, and the installation of OpenVino for model optimization was followed by the instructions provided on the official website. Each model was optimized using FP16, which provides the best optimization. However, it slightly decreased the overall accuracy. We used the latest Jetpack 5.0 for Jetson Nano and Xavier and optimized each model using FP16. On each edge platform, we created a separate virtual environment and installed all the required libraries and packages on it. We needed to swap memory during experimentation in all edge platforms. TensorRT was used to optimize the models on Jetson Xavier and Nano, and OpenVino for NCS2.

4.2. Evaluation Protocol

Mean average precision (mAP) is the key parameter for analyzing the accuracy of any detection system, and frames per second are essential for evaluating the time performance of any deep learning model. The efficiency of the detection models was compared using these two metrics. Three distinct IoU thresholds, namely 0.50:0.05:0.95, 0.75, and 0.5, were used to attain the mAP. The multiple IoU threshold of 0.50:0.05:0.95 has a geometric overlap between prediction and ground truth values for all categories and a 0.05 step size. This accumulated number is known as the mean AP (or mAP).

4.3. Results Discussion

We achieved results on a system having a discrete P5000 GPU and edge devices, i.e., NCS, Jetson Nano, and Xavier. The details of the results for the three challenging datasets are discussed below:

4.3.1. Discrete GPU-Based Detection

The standard mean AP via the validation set using GPU with respect to three datasets is shown in Table 2. It was observed that the modified CenterNet with hourglass-104 as a backbone achieved the highest mAP against each dataset, while Resnet18 had the lowest mAP. The highest mAP of 91.62%, 75.62%, and 34.82% are reported for AU-AIR, SDD, and VisDrone, respectively. We obtained 40.02 FPS with Resnet18 and 7.21 FPS with Res2net101. Other mAP and FPS for each backbone with respective datasets are shown in Table 2.

Table 2. mAP and FPS achieved for each dataset using the modified CenterNet and different CNN backbones with Val-Dev.

Backbones	**mAP**			**FPS**
	AU-AIR	**SDD**	**VisDrone**	
resnet18	70.38	43.38	19.05	**40.02**
resnet34	72.51	50.51	22.01	32.32
resnet50	77.52	52.52	24.03	30.19
DLA-34	80.31	64.31	25.90	28.91
resnet101	82.03	69.03	25.12	17.12
resnet152	86.37	71.37	28.17	16.07
res2net50	88.93	72.43	28.51	15.73
res2net101	90.52	74.52	34.43	7.21
hourglass-104	**91.62**	**75.62**	**34.82**	7.19

4.3.2. AI-on-the-Edge Based Detection

We optimized the models and compared the results on different edge platforms, i.e., NCS, Jetson Nano, and Xavier. It was observed that Res2net101 achieved the highest mAP against the respective dataset, whereas the highest FPS was achieved using ResNet18 as a backbone with modified CenterNet. We obtained 90.15%, 73.29%, and 33.12% mAP against AU-AIR, SDD, and VisDrone, respectively, on Jetson Xavier using Res2Net101 as a backbone. On Jetson Nano, we obtained 90.12% and 33.12% mAP against AU-AIR and VisDrone with res2net101 as a backbone, but 72.42% mAP for the SDD dataset and hourglass as a backbone. We found 90.21%, 73.89%, and 33.22% mAP for AU-AIR, SDD, and VisDrone, respectively, with Res2Net101 as a backbone. The results achieved using three datasets and different CNN backbones with respective edge platforms are shown in Table 3. It was observed that ResNet18 as a backbone with modified CenterNet had the highest FPS. Because these edge platforms have less computation power, the FPS was observed to be lower than the FPS achieved on discrete GPUs. Moreover, the acceptable FPS for a drone would be at least 25–30 according to its speed. Since the limited resource of the edge platforms makes drones not applicable to our detector, however, we may introduce the lightweight deep learning model by compromising the detection accuracy. We can also use more GPUs in each edge platform (i.e., Jetson Xavier, nano, and NCS2) to improve the detection speed or FPS for our approach.

Table 3. mAP and FPS achieved after optimization on edge platforms against three datasets via Val-Dev.

Model	Backbones	Jetson Xavier				Jetson Nano				NCS			
		AU-AIR	SDD	VisDrone	FPS	AU-AIR	SDD	VisDrone	FPS	AU-AIR	SDD	VisDrone	FPS
CenterNet	ResNet18	70.02	42.83	18.22	**7.54**	70.02	42.83	18.22	**4.50**	70.21	42.93	18.32	**4.34**
CenterNet	ResNet34	72.26	49.59	21.01	7.34	72.26	49.59	21.01	4.31	72.46	49.99	21.81	4.14
CenterNet	ResNet50	77.22	51.55	23.28	6.94	77.22	51.55	23.28	3.92	77.32	51.95	23.98	3.9
CenterNet	DLA-43	80.11	64.32	24.92	6.01	80.11	64.32	24.92	3.03	80.21	64.09	24.12	3.31
CenterNet	ResNet101	81.69	68.25	24.35	5.93	81.69	68.25	24.35	2.94	81.89	68.75	24.95	3.01
CenterNet	ResNet152	85.78	70.31	27.17	5.54	85.78	70.31	27.17	2.54	85.98	71.01	27.47	2.54
CenterNet	Res2Net50	88	71.32	27.29	4.41	88	71.32	27.29	2.45	88.03	71.72	27.69	2.41
CenterNet	Res2Net101	**90.15**	**73.29**	**33.12**	3.43	**90.12**	**73.29**	**33.12**	2.46	**90.21**	**73.89**	**33.22**	1.43
CenterNet	hourglass104	89.98	73.20	33.08	3.41	90.04	73.19	33.05	2.41	90.18	73.79	33.19	1.4
YoloV6	Efficientrep	87.93	72.02	31.01	3.94	89.06	69.91	32.69	3.89	89.05	72.72	32.86	2.34
YoloV7	RepConN	88.01	72.53	31.85	3.03	89.18	72.42	33.09	2.34	90.09	73.02	33.21	2.31

4.4. Ablation Study

Table 4 shows the comparison of the modified CenterNet versus SOTA detection models against the IoU threshold of 0.5 and 0.75, respectively. We implemented all versions of YOLO, SSD, and Mask-RCNN and validated them on the three challenging datasets. It was observed that the modified CenterNet with Res2net101 and hourglass-104 as backbones performed better as compared to other detection algorithms. After the modified CenterNet, YoloV7 achieved better results for the three datasets. In comparison, the modified CenterNet performs 1.27% and 2.0% better against AU-AIR, 4.88% and 9.37% against SDD, and 4.3% and 2.2% for the VisDrone dataset, for IoU thresholds of 0.5 and 0.75, respectively.

Table 4. Comparison of mAP achieved between state-of-the-art detection models vs. modified CenterNet for three datasets.

Model	Backbone	AU-AIR		SDD		VisDrone		FPS
		mAP@0.5	mAP@0.75	mAP@0.5	mAP@0.75	mAP@0.5	mAP@0.75	
CenterNet	ResNet18	72.93	69.63	59.93	41.63	40.63	21.76	**40.02**
CenterNet	ResNet34	61.91	71.43	61.91	47.43	50.65	24.43	32.32
CenterNet	ResNet50	79.33	76.92	66.33	49.92	52.78	25.91	30.19
CenterNet	DLA-43	83.02	79.82	70.42	61.82	53.04	27.76	28.91
CenterNet	ResNet101	85.32	80.13	74.32	66.01	54.90	29.03	17.12
CenterNet	ResNet152	88.72	84.31	79.72	69.31	55.56	29.67	16.07
CenterNet	Res2Net50	90.04	85.49	80.04	70.49	56.43	30.14	15.73
CenterNet	Res2Net101	91.31	**90.03**	**84.31**	72.43	**57.41**	31.65	7.21
CenterNet	hourglass-104	**92.20**	89.40	84.20	**74.40**	57.25	**35.55**	7.19
YoloV1	CNN	60.32	43.98	50.42	39.18	32.41	12.72	17.53
YoloV2	Darknet-19	66.32	50.38	55.32	41.38	34.21	13.22	16.32
YoloV3	Darknet-53	70.32	62.51	58.09	49.51	41.02	18.01	15.23
YoloV4	CSP-Darknet-53	73.03	67.52	68.03	52.52	42.21	20.28	12.62
YoloV5	FPN	80.42	76.92	71.43	60.43	44.32	28.92	12.17
YoloV6	Efficientrep	88.51	81.14	75.61	63.51	50.35	31.45	11.98
YoloV7	RepConN	90.93	88.03	79.43	65.03	53.11	33.35	11.32
SSD	MobileNetV2	72.82	68.37	74.98	43.38	43.32	20.17	32.12
Faster-RCNN	RPN	83.91	76.78	69.3	49.51	47.37	23.65	11.29

The modified CenterNet using hourglass-104 as a backbone achieved the best results. Therefore, Table 5 shows the class-wise average precision (AP) of each class of the three datasets. The bus class has the highest 94.21% AP in AU-AIR, the cart call has 57.79% AP in SSD, and the car class has 57.79% AP in the VisDrone dataset. Other classes and their achieved AP are also shown in the table.

Table 5. Class-wise AP of modified CenterNet using Hourglass-104 as backbones for their respective dataset.

Class	**AP**		
	VU-AIR	**SSD**	**VisDrone**
Pedestrian	-	32.62	32.62
Person/Human	90.43	-	16.23
Bicycle	80.82	-	11.43
Car	88.64	44.57	57.79
Van	91.43	-	44.57
Truck	89.31	-	35.99
Tricycle	-	-	35.81
Awn..	-	-	33.19
Bus	94.21	35.99	54.68
Motor/Bike/Motorbike	79.67	16.23	27.19
Skateboarder	-	11.43	-
Cart	-	57.79	-
Trailer	90.53	-	-

4.5. Qualitative Analysis

We compared the results achieved from the SOTA detection models (YoloV7 as a one-stage detector and Faster-RCNN as a two-stage detector) versus the modified CenterNet with ResNet18 and hourglass-104 as backbones. Figure 7 shows three rows of detection results against detection models and their respective datasets. Since the datasets were complex in nature, we took simple images (having no occlusion and visibility issues) and inferenced them via trained Mask-RCNN, YoloV7, and modified CenterNet to obtain a qualitative comparison. Second, these datasets have car, pedestrian, and motor instances in common; therefore, we performed an inference with these examples. Row 1 contains car and motor instances, and the best detection confidence was discovered using a modified CenterNet–Hourglass104. The white car had confidence scores of 0.8 via both Faster RCNN and YoloV7, whereas the modified CenterNet detected it with better confidence scores of 0.9. Similarly, other cars and one motor had better confidence scores using modified CenterNet–Hourglass104 than CenterNet–ResNet18. Row 2 shows the detection results for the SSD dataset. It was observed that the modified CenterNet detected pedestrians with better confidence scores than Mask-RCNN and YoloV7. Row 3 shows only car instances, and it is observed that the modified CenterNet detects red, black, and white cars with 1.0, 0.9, and 1.0 confidence scores, respectively, which are better than the Mask RCNN and YoloV7. It was also observed that the modified CenterNet with ResNet18 detected white cars with 0.9 confidence, which is less than the detection confidence via CenterNet–Hourglass104.

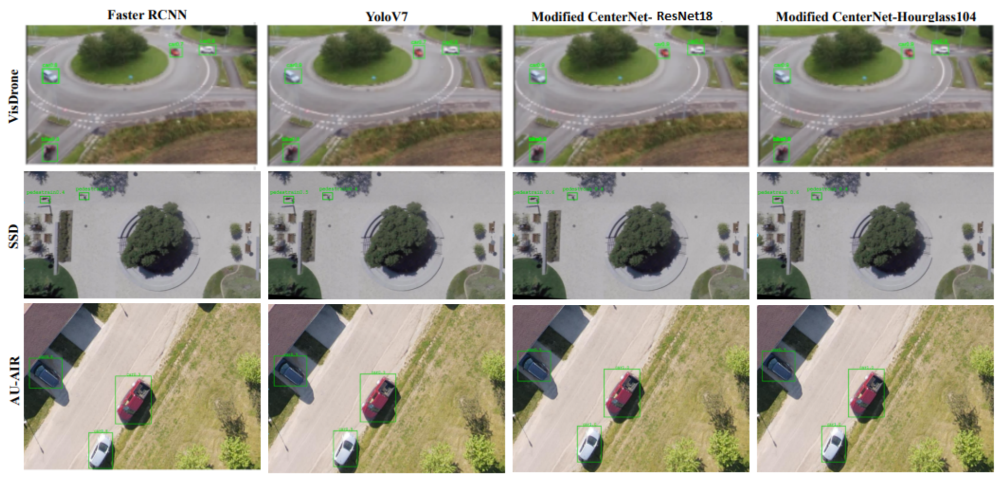

Figure 7. Qualitative comparison of SOTA detection algorithms vs our approach.

4.6. Comparison of SOTA Vs Our Proposed Approach

We compared our results with cutting-edge techniques in peer-reviewed publications. Table 6 depicts the comparison of mAP achieved using other deep learning methodologies for small-scale object detection versus our approach. Junfeng et al. used the VistrongerDet deep learning model and then validated it using the VisDrone dataset and achieved 31.85% mAP. However, our approach achieved 34.85% mAP, which is 3.0% greater than theirs. Similarly, Ammar et al. used YOLOV4 and achieved 74.81% mAP, but our approach is 0.81% better than those using the SSD dataset. We also compared our approach acquired by validating on the AU-AIR dataset. Bozean et al. achieved 91.2% mAP using the CADNet object detection model, but our approach performs 0.42% better than theirs. In a nutshell, our approach (validated with the three challenging datasets) performs better than the latest techniques.

Table 7 illustrates the comparison of frames per second (FPS) obtained using different GPU-based platforms. We compared the FPS achieved using the deep learning approach and found that our approach has better FPS for all the edge devices. Ref. [26] has eight FPS at Jetson Xavier, which is 0.46% less than ours.

Table 6. Quantitative analysis of SOTA vs. our approach.

Method	Dataset	mAP
Improved RetinaNet [27]	VisDrone	29.4
DPNet-Ensemble [28]	VisDrone	29.62
RetinaNet [29]	VisDrone	31.8
VistrongerDet [30]	VisDrone	33.85
Modified CenterNet–Hourglass-104	**VisDrone**	**34.85**
YOLO-V2 [31]	SSD	68.0
Deep Learning [7]	SSD	68.71
PeleeNet+DMS [32]	SSD	74.44
YOLO-V4 [33]	SSD	74.81
Modified CenterNet–Hourglass-104	**SSD**	**75.62**
YOLO-V4 [33]	AU-AIR	67.35
Multimodal CNN and LSTM [34]	AU-AIR	71.84
Visual Saliency [35]	AU-AIR	81.3
CADNet with Encoder/Decode [36]	AU-AIR	91.2
Modified CenterNet–Hourglass-104	**AU-AIR**	**91.62**

Table 7. Quantitative analysis of SOTA vs. our approach with respect to FPS.

Method	Discrete GPU	Xavier	Nano	NCS
Faster RCNN [37]	7.60	-	-	-
DSSD513 [38]	5.50	-	-	-
YOLOV2 [26]	11.3	2.9	-	1.8
SSD MobileNet [26]	14.09	8	-	5.5
LiteReconfig [39]	12	1.5	-	-
EfficientNet-b2 [40]	15	3.19	3.03	-
Modified CenterNet–ResNet18	40.02	7.54	4.50	4.34
Modified CenterNet–ResNet34	32.32	7.34	4.31	4.14
Modified CenterNet–ResNet50	30.19	6.94	3.92	3.9
Modified CenterNet–DLA-43	28.91	6.01	3.03	3.31
Modified CenterNet–ResNet101	17.12	5.93	2.94	3.01
Modified CenterNet–ResNet152	16.07	5.54	2.54	2.54
Modified CenterNet–Res2Net50	15.73	4.41	2.45	2.41
Modified CenterNet–Res2Net101	7.21	3.43	2.46	1.43
Modified CenterNet–Hourglass104	7.19	3.34	2.34	2.34

5. Conclusions

We modified CenterNet and provided a comparison with nine different CNN-based backbones. We implemented SOTA detection models and validated them using three open challenge datasets, i.e., VisDrone, SSD, and AU-AIR datasets, and provided a comparison with modified CenterNet. Results showed that the modified CenterNet with hourglass104 as a backbone achieved the best mAP, whereas CenterNet with ResNet18 obtained the best FPS compared to SOTA detection algorithms and techniques found in the cutting-edge latest research. We optimized the models and implemented them after optimization on edge platforms, i.e., Jetson Xavier, Jetson Nano, and NCS2, and found better performance. Our modified CenterNet with hourglass as a backbone achieved 91.62%, 75.61%, and 34.82% mAP via AU-AIR, SSD, and VisDrone datasets, respectively; however, 40.02 FPS was achieved using the backbone of ResNet18. We also compared our approach with the latest cutting-edge research and found promising results for both discrete GPU and edge platforms. As for future work, we plan to modify other SOTA-detection models and implement them on edge devices after optimization. We can explore other optimization techniques to speed up the inference time for better detection.

Author Contributions: Conceptualization, Z.S. and M.H.Y.; Methodology, Z.S. and M.H.Y.; Software, Z.S.; Validation, Z.S., R.A. and S.A.V.; Formal Analysis, Z.S., R.A. and M.H.Y.; Investigation, Z.S.; Resources, M.H.Y.; Data Curation, Z.S. and R.A.; Writing—Original Draft Preparation, Z.S.; Writing—Review & Editing, M.H.Y., R.A., S.A.V. and S.V.; Visualization, S.V.; Supervision, M.H.Y., R.A., S.A.V. and S.V.; Project Administration, M.H.Y.; Funding Acquisition, M.H.Y. and S.V. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the Higher Education Commission of Pakistan funding for this project through a grant titled "Establishment of National Centre of Robotics and Automation" under Grant No. DF-1009-31 for this research work.

Data Availability Statement: The datasets are publicly available and accessible i.e., VisDrone http://aiskyeye.com/download/, Stanford Drone Dataset (SDD) https://cvgl.stanford.edu/projects/uav_data/ and AU-AIR https://bozcani.github.io/auairdataset, accessed on 1 March 2023.

Acknowledgments: We acknowledge the National Centre of Robotics and Automation (NCRA) Pakistan and University of KwaZulu-Natal for their consistent support.

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

References

1. Abbas, S.M.; Singh, S.N. Region-based object detection and classification using faster R-CNN. In Proceedings of the 2018 4th International Conference on Computational Intelligence Communication Technology (CICT), Ghaziabad, India, 9–10 February 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
2. Chen, Y.; Zhu, X.; Li, Y.; Wei, Y.; Ye, L. Enhanced semantic feature pyramid network for small object detection. *Signal Process. Image Commun.* **2023**, *113*, 116919. [CrossRef]
3. Jung, H.-K.; Choi, G.-S. Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions. *Appl. Sci.* **2022**, *12*, 7255. [CrossRef]
4. Saeed, Z.; Awan, M.N.M.; Yousaf, M.H. A Robust Approach for Small-Scale Object Detection From Aerial-View. In Proceedings of the 2022 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Sydney, Australia, 30 November–2 December 2022; pp. 1–7. [CrossRef]
5. Jawaharlalnehru, A.; Sambandham, T.; Sekar, V.; Ravikumar, D.; Loganathan, V.; Kannadasan, R.; Khan, A.A.; Wechtaisong, C.; Haq, M.A.; Alhussen, A.; et al. Target Object Detection from Unmanned Aerial Vehicle (UAV) Images Based on Improved YOLO Algorithm. *Electronics* **2022**, *11*, 2343. [CrossRef]
6. Wang, K.; Wei, Z. YOLO V4 with hybrid dilated convolution attention module for object detection in the aerial dataset. *Int. Remote. Sens.* **2022**, *43*, 1323–1344. [CrossRef]
7. Maktab Dar Oghaz, M.; Razaak, M.; Remagnino, P. Enhanced Single Shot Small Object Detector for Aerial Imagery Using Super-Resolution, Feature Fusion and Deconvolution. *Sensors* **2022**, *22*, 4339. [CrossRef] [PubMed]
8. Xia, G.-S.; Bai, X.; Ding, J.; Zhu, Z.; Belongie, S.; Luo, J.; Datcu, M.; Pelillo, M.; Zhang, L. DOTA: A large-scale dataset for object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3974–3983.

9. Ding, J.; Xue, N.; Long, Y.; Xia, G.; Lu, Q. Learning roi transformer for oriented object detection in aerial images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2849–2858. [CrossRef]
10. Liu, Z.; Wang, H.; Weng, L.; Yang, Y. Ship rotated bounding box space for ship extraction from high-resolution optical satellite images with complex backgrounds. *IEEE Geosci. Remote. Sens. Lett.* **2016**, *13*, 1074–1078. [CrossRef]
11. Ma, J.; Shao, W.; Ye, H.; Wang, L.; Wang, H.; Zheng, Y.; Xue, X. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Trans. Multimed.* **2018**, *20*, 3111–3122. [CrossRef]
12. Yang, X.; Yang, J.; Yan, J.; Zhang, Y.; Zhang, T.; Guo, Z.; Sun, X.; Fu, K. SCRDet: Towards more robust detection for small, cluttered and rotated objects. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 8232–8241.
13. Xu, Y.; Fu, M.; Wang, Q.; Wang, Y.; Chen, K.; Xia, G.; Bai, X. Gliding vertex on the horizontal bounding box for multi-oriented object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1452–1459. [CrossRef] [PubMed]
14. Shams, S.; Platania, R.; Lee, K.; Park, S.J. Evaluation of deep learning frameworks over different HPC architectures. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 1389–1396.
15. Süzen, A.A.; Duman, B.; Şen, B. Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn. In Proceedings of the 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 26–28 June 2020; pp. 1–5.
16. Baller, S.P.; Jindal, A.; Chadha, M.; Gerndt, M. DeepEdgeBench: Benchmarking Deep Neural Networks on Edge Devices. In Proceedings of the 2021 IEEE International Conference on Cloud Engineering (IC2E), San Francisco, CA, USA, 4–8 October 2021; pp. 20–30.
17. Verma, G.; Gupta, Y.; Malik, A.M.; Chapman, B. Performance evaluation of deep learning compilers for edge inference. In Proceedings of the 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Portland, OR, USA, 17–21 June 2021; pp. 858–865. [CrossRef]
18. Cao, Y.; He, Z.; Wang, L.; Wang, W.; Yuan, Y.; Zhang, D.; Zhang, J.; Zhu, P.; Gool, L.V.; Han, J.; et al. VisDrone-DET2021: The vision meets drone object detection challenge results. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021.
19. Robicquet, A.; Sadeghian, A.; Alahi, A.; Savarese, S. Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016.
20. Bozcan, I.; Kayacan, E. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020. [CrossRef]
21. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
22. Joseph, R.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
23. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. [CrossRef]
24. Yu, F.; Wang, D.; Shelhamer, E.; Darrell, T. Deep layer aggregation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2403–2412. [CrossRef]
25. Hei, L.; Deng, J. Cornernet: Detecting objects as paired key- points. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018. [CrossRef]
26. Hossain, S.; Lee, D.-J. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. *Sensors* **2019**, *19*, 3371. [CrossRef] [PubMed]
27. Li, C.; Yang, T.; Zhu, S.; Chen, C.; Guan, S. Density map guided object detection in aerial images. proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. *arXiv* **2020**, arXiv:2004.05520.
28. Samyal, A.S.; Akshatha, K.R.; Hans, S.; Karunakar, A.K.; Satish Shenoy, B. Analysis and adaptation of yolov4 for object detection in aerial images. *arXiv* **2022**, arXiv:2203.10194.
29. Raza, M.A.; Naeem, H.B.; Yasin, A.; Yousaf, M.H. Birdview retina-net: Small-scale object detector for unmanned aerial vehicles. In Proceedings of the 2021 16th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 22–23 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
30. Wan, J.; Zhang, B.; Zhao, Y.; Du, Y.; Tong, Z. VistrongerDet: Stronger Visual Information for Object Detection in VisDrone Images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021.
31. Tang, T.; Deng, Z.; Zhou, S.; Lei, L.; Zou, H. Fast vehicle detection in uav images. In Proceedings of the 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP), Shanghai, China, 18–21 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–5. [CrossRef]
32. Jing, Z.; Xi, L.; Meng, W.; Liheng, Y.; Li, Z. Coarse-to-fine object detection in unmanned aerial vehicle imagery using lightweight convolutional neural network and deep motion saliency. *Neurocomputing* **2020**, *398*, 555–565. [CrossRef]

33. Ammar, A.; Koubaa, A.; Ahmed, M.; Saad, A.; Benjdira, B. Vehicle detection from aerial images using deep learning: A comparative study. *Electronics* **2021**, *10*, 820. [CrossRef]
34. He, M.; Hohil, M.; LaPeruta, T.; Nashed, K.; Lawrence, V.; Yao, Y.-D. Performance evaluation of multimodal deep learning: Object identification using uav dataset. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*; SPIE: Bellingham, WA, USA, 2021; Volume 11746, pp. 602–608.
35. Gupta, H.; Verma, O.P. Monitoring and surveillance of urban road traffic using low altitude drone images: A deep learning approach. *Multimed. Tools Appl.* **2022**, *81*, 19683–19703. [CrossRef]
36. Bozcan, I.; Kayacan, E. Context-dependent anomaly detection for low altitude traffic surveillance. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 224–230. [CrossRef]
37. 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 (NIPS), Montreal, QC, Canada, 7–12 December 2015; pp. 91–99. [CrossRef]
38. Fu, C.-Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. Dssd: Deconvolutional single shot detector. *arXiv* **2017**, arXiv:1701.06659.
39. Xu, R.; Lee, J.; Wang, P.; Bagchi, S.; Li, Y.; Chaterji, S. LiteReconfig: Cost and content aware reconfiguration of video object detection systems for mobile GPUs. In Proceedings of the Seventeenth European Conference on Computer Systems, Rennes, France, 5–8 April 2022. [CrossRef]
40. Tergel, M.-O.; Shenoy, R. Energy and Cost Consider- ations for GPU Accelerated AI Inference Workloads. In Proceedings of the 2021 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, USA, 8–10 October 2021; IEEE: Piscataway, NJ, USA, 2021. [CrossRef]