applied sciences

MDPI

Article

Fast and Robust People Detection in RGB Images

Florin Dumitrescu, Costin-Anton Boiangiu * and Mihai-Lucian Voncilă

Computer Science and Engineering Department, Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independenței 313, 060042 Bucharest, Romania; florin.dumitrescu97@upb.ro (F.D.); mihai_lucian.voncila@stud.acs.pub.ro (M.-L.V.)

* Correspondence: costin.boiangiu@cs.pub.ro

Abstract: People detection in images has many uses today, ranging from face detection algorithms used by social networks to help the users tag other people, to surveillance systems that can create a statistic of the population density in an area, or identify a suspect, or even in the automotive industry as part of the Pedestrian Crash Avoidance Mitigation (PCAM) system. This work focuses on creating a fast and reliable object detection algorithm that will be trained on scenes that depict people in an indoor environment, starting from an existing state-of-the-art approach. The proposed method improves upon the You Only Look Once version 4 (YOLOv4) network by adding a region of interest classification and regression branch such as Faster R-CNN's head. The candidate bounding boxes proposed by YOLOv4 are ranked based on their confidence score, the best candidates being kept and sent as input to the Faster Region-Based Convolutional Neural Network (R-CNN) head. To keep only the best detections, non-maximum suppression is applied to all proposals. This decreases the number of false-positive candidate bounding boxes, the low-confidence detections of the regression and classification branch being eliminated by the detections of YOLOv4 and vice versa in the non-maximum suppression step. This method can be used as the object detection algorithm in an image-based people tracking system, namely Tracktor, having a higher inference speed than Faster R-CNN. Our proposed method manages to achieve an overall accuracy of 95% and an inference time of 22 ms.

Keywords: convolutional neural networks; deep neural networks; single-stage object detector; people detection; regression; classification; Mobility Aids; MOT17Det; YOLO; R-CNN

Citation: Dumitrescu, F.; Boiangiu, C.-A.; Voncilă, M.-L. Fast and Robust People Detection in RGB Images. *Appl. Sci.* **2022**, *12*, 1225. https://doi.org/10.3390/app12031225

Academic Editors: Peng-Yeng Yin and Seokwon Yeom

Received: 25 October 2021
Accepted: 21 January 2022
Published: 24 January 2022

1. Introduction

Object detection is a computer vision task whose objective is to find certain objects of interest in an image and assign a bounding box and a category to them. Early object detection algorithms, such as the one proposed by Viola and Jones [1], were used to identify preset features (Haar attributes in this case) in an image, which helped with regression and classification. Starting with R-CNN [2], the deep neural network approach became popular among researchers, with most state-of-the-art object detection algorithms using this paradigm nowadays. However, even if a deeper neural network performs better theoretically, in practice the maximum depth is limited due to vanishing gradients. This problem was addressed by He et al. [3] with the introduction of residual layers, which allowed the networks to be much deeper by facilitating the propagation of the gradient throughout the network. The network the authors proposed, ResNet, is used today as the backbone of many state-of-the-art detectors.

Different object detection problems require different kinds of algorithms. If accuracy is the main target, then two-stage detectors such as the R-CNN family [2,4–10] are the better choice. A two-stage detector consists of a region proposal stage and a classification and regression stage, hence the name. Thanks to the region proposal, the detections tend to be sparse, and the algorithm can identify the objects better. On the other hand, if frequent detections are necessary, then a single-stage detector [11–26] is better since the two-stage

ones tend to be rather slow. They sacrifice accuracy for detection frequency by skipping the region proposal stage and detecting the objects directly on the feature maps. Because of this, the predictions of the network tend to be dense, and this can cause the network to misclassify an object or omit it altogether. More recent research on dense object detectors (single-stage detectors) managed to improve their accuracy while maintaining a decent detection frequency, closing the gap with two-stage detectors.

Other than the aforementioned methods, there have been many different approaches and researches conducted in recent years in the domain of object detection, amongst which are salient-based methods, such as the one in [27]. These are not of primary interest to our paper, but they have been analyzed and described in other literature reviews, such as the ones in [28–33].

The focus of this paper is to create an algorithm that uses both object detection paradigms and single-stage and two-stage object detectors. Our method aims to combine both these approaches in order to obtain higher accuracy when compared to other, similar methods at similar inference times. To achieve this, starting from the implementation of YOLOv4 [14], a regression and classification head is added, along with a region of interest pooling (ROI) layer, like that of Faster R-CNN [4]. The high-confidence candidates that do not overlap (i.e., IoU < threshold) with those outputted by YOLOv4 are then sent as input to the ROI pooling layer, which aligns the bounding boxes with the feature maps, each proposal being then regressed and classified. Finally, the concatenation of the proposals of both the regression and classification head and YOLOv4 is filtered by using non-maximum suppression. This method reduces the number of false-positive detections, as only the best proposals of either YOLOv4 or the regression and classification head are kept after filtering. Another benefit of this method is that, due to the regression and classification head, this model can be used in a "people tracking by detection" framework, acting both as a detector, as well as regressing the position of people in the current frame based on their position in the previous frame (see Bergmann et al. [34]).

This article is divided into seven sections that are arranged as follows: The Section 1 is the Introduction, which provides details regarding the importance of people detection in RGB images as well as our proposed method and contributions. The Section 2 is Related Work, which describes some of the more common approaches in the field, as well as presenting some of the more recent approaches in the past years. The Section 3 is Used Datasets and Training Tricks and details some approaches one could take when training their CNNs to improve the results, as well as the datasets we considered for our research. The Section 4 is Materials and Methods, which describes the architecture we implemented as well as some of the steps we took to ensure our model was properly trained. The Section 5 is Results, in which we present the results we obtained with our datasets as well as some of the metrics involved, such as precision, recall, and mean-average precision (mAP). The Section 6 is Discussions, which analyzes the results we obtained as well as presenting the main benefits and detriments of our approach when compared to other state-of-the-art approaches. The Section 7 is Conclusions and Future Directions, which summarizes what was achieved in this paper, as well as presenting some of the future work that could be performed regarding improving our method.

2. Related Work

2.1. Conventional Approaches

Even though most of the current object detection algorithms use deep learning to fulfill their task, earlier methods were heavily dependent on handcrafted features. One such early object detector is the Viola and Jones [1] algorithm, which uses Haar-like functions to represent features in an image. Using a sliding window approach and comparing the pixels in that region to the feature templates is very computationally expensive. Because of this, the authors proposed the usage of a technique called Integral Image, which involves pre-computing a map of the image starting from the top-right corner as follows: the value of each cell $p'_{i,j}$ is equal to the sum of the values of the cells directly above and to the

left as well as the value of the pixel $p_{i,\,j}$ With this approach, the sum of the pixels in a rectangular area can be quickly computed as $S_{rect} = p'_{b,\,r} + p'_{t,\,l} - \left(p'_{t,\,r} + p'_{b,\,l}\right)$, where t = top, b = bottom, l = left, and r = right, making it easier to compare with the templates by computing the different sums of the rectangular shapes composing them. Examples of such templates can be seen in Figure 1. The feature selection and matching are performed using AdaBoost (introduced in Schapire [35]), which automatically finds the best Haar-like filters to use from a pool of about 180 k templates. However, using the entire set of templates is computationally expensive. Because of this, the authors proposed the implementation of a cascade of weaker classifiers, each subsequent classifier testing an increasing number of features compared to the previous one. Each window is passed through the cascade network and, if a classifier in the chain considers that no object of interest is present, the algorithm discards the proposal. In this way, the algorithm manages to decrease the inference time per image, as many proposals are discarded in the earlier stages of the cascade network. Figure 2 depicts the flow schematic of the cascade network.

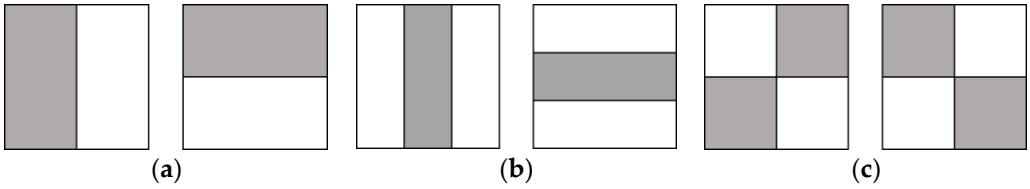


Figure 1. Examples of the rectangle (Haar-like) features. (**a**) Two-rectangle features. (**b**) Three-rectangle features. (**c**) Four-rectangle features.

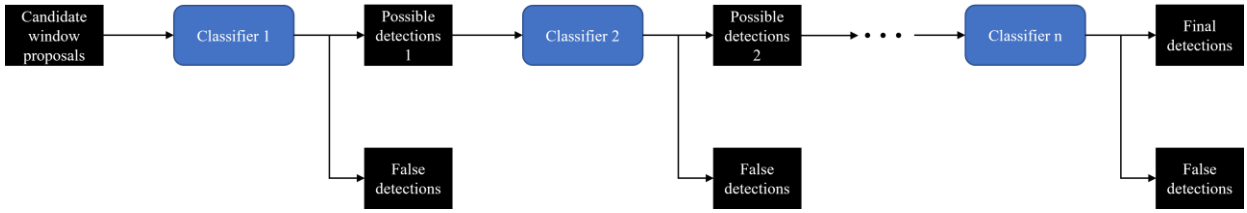


Figure 2. The schematic of the cascade network proposed by Viola and Jones [1].

"Histogram of Oriented Gradients for Human Detection", another early approach proposed by Dalal and Triggs [36], achieved robustness towards local illumination and small geometric transformations by introducing histogram of oriented gradients (HOG) descriptors, which model an object's shape and appearance by using the gradients of pixel intensities. These are computed regarding a sliding window as follows:

1. The entire image is color and gamma normalized to be independent of the color space.
2. The detection window is divided into a grid.
3. For each cell in the grid:
   a. The gradients are computed in each pixel;
   b. A histogram of the gradients in the cell is computed.
4. The cells are grouped into overlapping blocks.
5. Using the individual cell histograms, a histogram is computed for each block.
6. Compute the histogram of gradients over the blocks, using the histograms of the cells.
7. Normalize the block histograms to obtain illumination invariance.
8. Concatenate the block histograms to obtain the window's HOG descriptor.

The final step is using a classifier over the window's HOG descriptor. In the paper, the authors used a linear support vector machine (SVM) [37] for this task.

2.2. Deep Learning Approaches

2.2.1. Two-Stage Object Detection Algorithms

The two-stage detectors first propose regions of interest (ROI), which are then fed to a regressor and a classifier. The most notable algorithms in this category are the ones from the R-CNN family. The first of its kind is R-CNN [2], which uses selective search [38] for ROI proposal given an input image, and a pre-trained CNN that is used to extract the features of each ROI. Finally, an SVM and a regressor are used to output the bounding boxes and the probabilities of them containing an object. Figure 3a shows the flow of this algorithm.

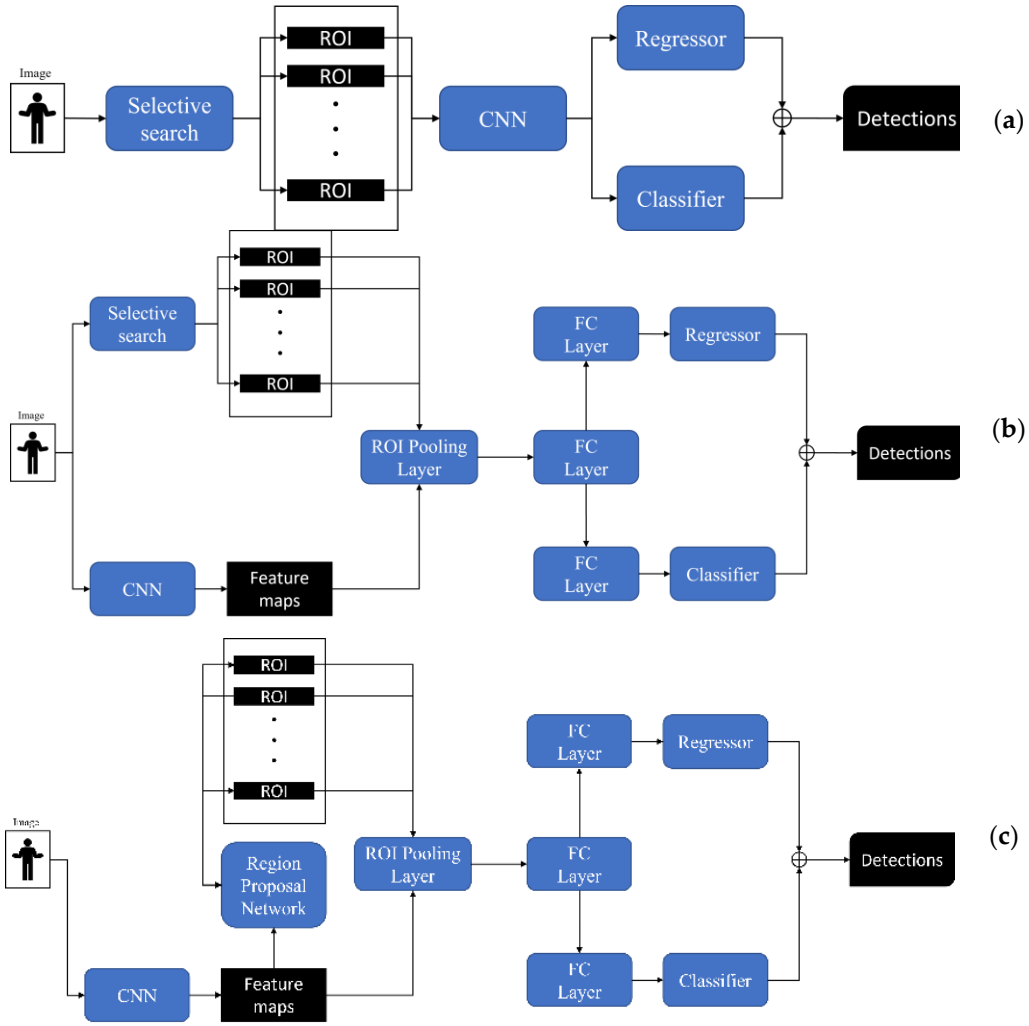


Figure 3. Network flows schematics (**a**) R-CNN, (**b**) Fast R-CNN, (**c**) Faster R-CNN.

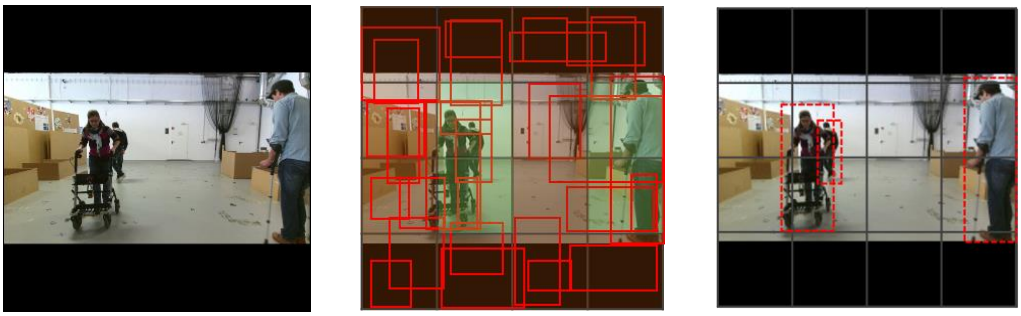
The main drawback of R-CNN is that it has a very high inference time, as each ROI is fed to the CNN independently. To overcome this aspect, Girshick [4] proposed an improved version called Fast R-CNN. Instead of feeding each ROI to the CNN, Fast R-CNN extracts the feature maps of the entire input image at once. The ROI proposal method remains the same, but a new ROI pooling layer is introduced, which divides the ROI into a grid of fixed size (the number of cells in the grid is independent of the size of the ROI) and then applies max pooling for each cell. The output of this layer is then sent to Fully Connected (FC) layers to predict the bounding box and the class of the object. Figure 3b shows the flow of this algorithm.

Faster R-CNN [5] further improved the performance of this method by introducing the Region Proposal Network (RPN), which uses anchor boxes (see Appendix A.1) and a sliding window to predict ROIs. The advantage of this method is that it can be trained to

predict more accurate bounding boxes, thus resulting in fewer low-quality predictions and a lower inference time per image. Figure 3c shows the flow of this algorithm.

2.2.2. Single-Stage Object Detection Algorithms

Unlike two-stage detectors, single-stage object detection algorithms do not require a separate region proposal stage, the ROI proposal, regression, and classification being performed all at once. Because of this, single-stage detectors are usually faster than their two-stage counterparts, but at the cost of accuracy. However, more recent algorithms in this category have managed to bridge the gap in terms of accuracy while still maintaining their low inference time. A pioneering single-stage object detector is YOLO [11], which works by dividing the image into an S × S grid and proposing B bounding boxes for each grid cell. If an object's center is inside a cell, that cell is said to contain the object. Separate from the proposed bounding boxes, each grid cell will have associated a class confidence score. All predictions are performed using only the features of the objects without relying on an anchor box. An illustration of how YOLO works can be seen in Figure 4. Later iterations [12–14,39] have adapted the anchor box concept, as well as introduced other improvements to increase the reliability of the detector. Its latest iteration, YOLOv4 [39], is currently among the best-performing algorithms in terms of average precision (AP) over the COCO dataset, according to [40].



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

Figure 4. Example of how YOLOv1 works. (**a**) The original padded image. (**b**) The bounding box proposals and the class probabilities associated with each cell. The green cells contain a person, while the orange ones do not. (**c**) The final detections.

Another well-known single-stage detection algorithm is Single Shot MultiBox Detector (SSD) [16] which, like YOLO, divides the image into an S × S grid, but innovates by using k anchor boxes in each grid cell. In addition to the anchor boxes, it also uses a Feature Pyramid Network (FPN) [41] to capture objects at different scales. According to their paper, SSD512 performs better than Faster R-CNN on the COCO test-dev2015 dataset, with a mAP@0.5 of 46.5, 1.2% better than Faster R-CNN, and mAP@0.75 of 27.8% compared to 23.5%, while also being faster. Advancements in backbone architectures have further improved these algorithms in both reliability and speed.

A common problem among all single-stage object detectors is a class imbalance, as most of the candidate locations do not contain an object, which affects the training performance as more predictions are processed and the easy negatives outweigh the other candidates. This problem is addressed in two-stage detectors by the RPN in the case of Faster R-CNN (e.g., by using hard negative mining), which filters the easy negatives. To address this issue in the case of single-stage detectors, Lin et al. [42] proposed a new loss function based on binary cross-entropy (BCE), called focal loss. In this paper, the authors also introduced a new network called RetinaNet, which uses an FPN to sample candidates at different scales and classification and regression subnetworks attached to every pyramid level. The candidates are selected using anchor boxes, which makes it an anchor-based method.

Almost all the previous one-shot detectors covered here were anchor-based. However, anchors are usually tailored to a certain dataset, which means that they introduce additional hyperparameters that need to be fine-tuned. Tian et al. [43] proposed a novel anchor-free method, FCOS, which computes an offset between each location (x, y) on a feature map and the coordinates of the top, bottom, left, and right side of the target bounding box. A location (x, y) can have multiple associated targets, but only the one with the smallest area will be considered. Because this greedy approach can lead to detections that have a small Intersection over Union (IoU), see Appendix A.2, with the target, a new metric called 'center-ness' is proposed, which computes the distance between the candidate bounding box center and the target center and helps filter these detections in the Non-Maximum Suppression (NMS) step during inference, as well as during training as an additional loss.

YOLO, SSD, RetinaNet, and FCOS are all center-based detectors, as they use the center point of an object to define positive samples, either by using anchors or not, and then regress the bounding boxes. Keypoint-based detectors, on the other hand, learn to detect certain points of a bounding box and then regress the rest. For example, CornerNet [44] predicts the top-left and bottom-right points of a bounding box, CenterNet [45] further improves by also exploring the center of the bounding box, while the algorithm proposed by Zhou et al. [46] learns to detect the center of the objects and then regresses all the other properties.

3. Used Datasets and Training Tricks

This category contains methods that improve the model's accuracy without increasing the inference time during testing, which is possible thanks to the training phase being performed offline, as well as the datasets one might consider when training such a model, which are presented in Section 3.1. Presented in Sections 3.2 and 3.3 are the gradient descent algorithms and the loss functions that are used in the experiments, these two being interconnected. Their role is to ensure that the model will converge in a local minimum point that is as close to the global minimum as possible. As such, depending on the task, an algorithm (loss function or gradient descent optimization) might lead to a better performance of the model, or cause it to not be able to learn (e.g., exploding gradients). In the case of exploding gradients, there are additional methods that help mitigate the problem, such as gradient clipping or gradient scaling, but these are not covered in this paper. Lastly, Section 3.4 presents a non-exhaustive list of data augmentation methods that are used in this paper during the training phase. As machine learning models are extremely dependent on the data they are trained onto, data augmentation methods are important for introducing diversity in the dataset, which in turn helps the model better generalize, especially on small datasets.

3.1. Existing Datasets

The first considered dataset is Mobility Aids, which was created and used by Vasquez et al. [47]. It contains roughly 17,000 RGB, Depth, and DepthJet images grouped into sequences. The images are split into a training set containing around 11,000 images and two test sets, the first one having around 4000 images and being used for testing the object detection algorithm. The rest of the images (1801), grouped into four sequences, form the second test set, which is used exclusively for testing tracking algorithms, as it contains annotations for both visible and occluded people. The dataset features five different 'objects' of interest that are represented by people with different disabilities (1. person without any handicap; 2. person using crutches; 3. person in a wheelchair; 4. person in a wheelchair that is pushed by another person; 5. person using a walking frame). The sequences are recorded at 15 frames per second (FPS) using both a fixed and a mobile camera placed on a robotic platform at eye level. Figure 5 shows image samples from this dataset. The annotations contain information about the position of the top-left and bottom-right corners of the bounding boxes, the class of the person, and the depth.

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

Figure 5. Samples from the Mobility Aids dataset [47]. (**a**) Scene from a large room with various actors walking around. (**b**) Scene of a narrow hallway featuring occlusion and people who are easy to misclassify.

Another dataset that features images recorded at eye level is MOT17Det [48–50], which features 14 sequences of images (for a total of 11,235 frames) split into training and test sets at a ratio of around 50–50%. MOT15 and MOT16 have also been considered, but the sequences of interest in MOT15 are already contained in this dataset and, as stated in Milan et al. [50], MOT17 contains all the sequences of MOT16 with more accurate ground truth. Out of the 14 sequences, only seven contain ground truths (which are necessary for training/testing), two of which feature an indoor (or close to indoor) environment. This means that five of the seven sequences with annotations can be used for training and two for testing, MOT17-09 and MOT17-11, both of which were recorded at 30 FPS with a camera positioned at eye-level; however, while MOT17-09 uses a static camera, MOT17-11 has a mobile one. All MOT_Challenge datasets have a single 'object' of interest and that is people. The annotations contain information about the bounding boxes (top-left corner, width, and height), the identity of the detected person, and a visibility ratio in case of occlusion. Samples from this dataset can be seen in Figure 6. The difference between MOT17 and MOT17Det is that MOT17Det also provides the ground truth annotations as opposed to MOT17, whose annotations are the output of an object detection algorithm. This makes this dataset suitable for training and testing a people detection algorithm.

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

Figure 6. Samples from the MOT17Det dataset [48–50]. (**a**) Sample from MOT17-09 that contains images taken from outside a shop alley. (**b**) Sample from MOT17-11 which contains images taken from inside a shopping mall.

Both datasets have been converted to COCO [51] format to use the same data loader.

3.2. Gradient Descent Optimization Algorithms

Because the data are iteratively fed to the algorithm in batches instead of all at once, a gradient optimization method must be employed for the algorithm to be able to learn. Perhaps one of the most popular and simplest algorithms in this category is the Stochastic Gradient Descent, which uses the gradient and a parameter called learning rate to update the weights of the model. The equation of this method is shown below:

$$\theta_t = \theta_{t-1} - \eta \, g_t, \tag{1}$$

where θ_t is the weight to be updated, η is the learning rate and g_t is the gradient computed at the current step.

Further advancements have been made to this algorithm, such as the addition of momentum, which smoothens the gradient when close to a local optimum point, or the Nesterov accelerated gradient, which is a momentum-based gradient method that also considers where the parameter would be after the update and adapts accordingly.

There are also adaptive gradient methods, a frequently used one being Adam [52]. However, as noted by Loshchilov et al. [53], the weight decay regularization employed is not equivalent to the L2 regularization as many people think, but is modified during the optimization steps. To solve this problem for the adaptive methods, Loshchilov et al. [53] proposed AdamW, in which the weight decay step is decoupled from the optimization step. Algorithm 1 shows the differences between the two methods, whilst Table 1 shows the meaning of the parameters used, as well as the default values they are initialized with.

Algorithm 1: Pseudocode Adapted from [1,54]. The Code Written in Red is Used only by Adam with L2 Regularization, While the One Written in Green Is only for AdamW.

1. Repeat until stopping criterion is met:
   1. Increment step time: $t \leftarrow t + 1$
   2. Update gradients: $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda\theta_{t-1}$
   3. Compute the first moment vector: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$
   4. Compute the second moment vector: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$
   5. Correct first moment bias: $\hat{m}_t \leftarrow m_t / \left(1 - \beta_1^t\right)$
   6. Correct second moment bias: $\hat{v}_t \leftarrow v_t / \left(1 - \beta_2^t\right)$
   7. Update learning rate: $\eta_t \leftarrow SetScheduleMultiplier(t)$
   8. Update parameters: $\theta_t \leftarrow \theta_{t-1} - \eta_t\left(\alpha\hat{m}_t / \left(\sqrt{\hat{v}_t} + \epsilon\right) + \lambda\theta_{t-1}\right)$

Table 1. Parameters used by Adam with L2 regularization and AdamW.

Parameter	Type	Value	Meaning
α	input	0.001	learning rate
β_1	input	0.9	first moment estimates decay rate
β_2	input	0.999	second moment estimates decay rate
λ	input	0.01	parameters vector decay rate
t	-	0	current time step
m_t	-	$m_{t=0} = 0$	first moment estimates vector
v_t	-	$v_{t=0} = 0$	second moment estimates vector
η_t	input	computed	learning rate schedule multiplier
θ_t	output	$\theta_{t=0}$ = random values	optimized parameter vector

3.3. Loss Functions

The loss functions are used by machine learning algorithms to compare the difference between the solution proposed by the model and the ground truth. In this paper, two loss functions are used: generalized intersection over union and focal loss.

The *IoU* loss L_{IoU} used for the regression task, computes the dissimilarity between the proposed bounding box and the ground truth bounding box as $L_{IoU} = 1 - IoU$. However, this method does not work properly when the two bounding boxes do not intersect ($IoU = 0$), as it does not consider the distance between them. To solve this, Rezatofighi et al. [54] proposed the generalized intersection over union loss function ($L_{GIoU} = 1 - GIoU$), which uses a minimum surface rectangle C that encloses both bounding boxes to compute the relative distance between them. The equation for the generalized intersection over union ($GIoU$) is:

$$GIoU = IoU - \frac{A_C - A_U}{A_C}, \tag{2}$$

where A_C is the area of the rectangle C and A_U is the area of the union of the two bounding boxes. Unlike IoU, $GIoU$ outputs values in the range (−1,1), the negative values being

obtained when $IoU = 0$, which causes the loss to increase the further the two bounding boxes are from each other.

For the classification task, focal loss [42] is one of the most popular choices, as it alleviates the problems caused by a class imbalance in an image (foreground vs. background), especially in the case of single-stage object detectors, and is defined as:

$$FL(p_t) = -\alpha_t(1 - p_t)^{\gamma} log(p_t), \quad (3)$$

where:

- p_t is the confidence of the model that the bounding box contains an object of class t;
- α_t is the weighting factor for class t;
- γ is the focusing parameter.

It addresses the problem of class imbalance by introducing the scaling factor $(1 - p_t)^{\gamma}$, which increases the impact of misclassified proposals (hard negatives) on the total loss, while also decreasing the contribution of correctly classified examples (easy negatives or true positives).

3.4. Data Augmentation Methods

The dataset is usually limited and iterating over it for multiple epochs would lead to the model overfitting the data. Because of this, data augmentation methods are usually employed to help the model better generalize. There are multiple ways in which this can be done, but for simplicity, they will be grouped into five categories:

- Geometric transformations: scaling, left-right or upside-down flip, rotation, shear. These transformations must also be done on the ground truth annotations;
- Color space transformations: Hue-Saturation-Value transformations, random noise;
- Kernel filters: Gaussian-blur, sharpening;
- Data loss: Random erasing [55], Dropout [56], CutOut [57], etc.;
- Mixing images: CutMix [58], MixUp [59], Mosaic [60], etc.

The first three categories are straightforward. Random erasing overwrites patches from the image with random data or a set value. CutMix works the same as CutOut, but instead of a set value, it uses another image. DropOut works at the layer level, each layer having a probability to drop some of its output. MixUp interpolates two images. In the case of both CutMix and MixUp, the labels are modified to include both the classes from the original image as well as the classes from the additional image used in the process. Mosaic concatenates 4 images, the annotations of each image being offset to match the position of the image. Examples of CutOut, CutMix, MixUp, and Mosaic can be seen in Figure 7.

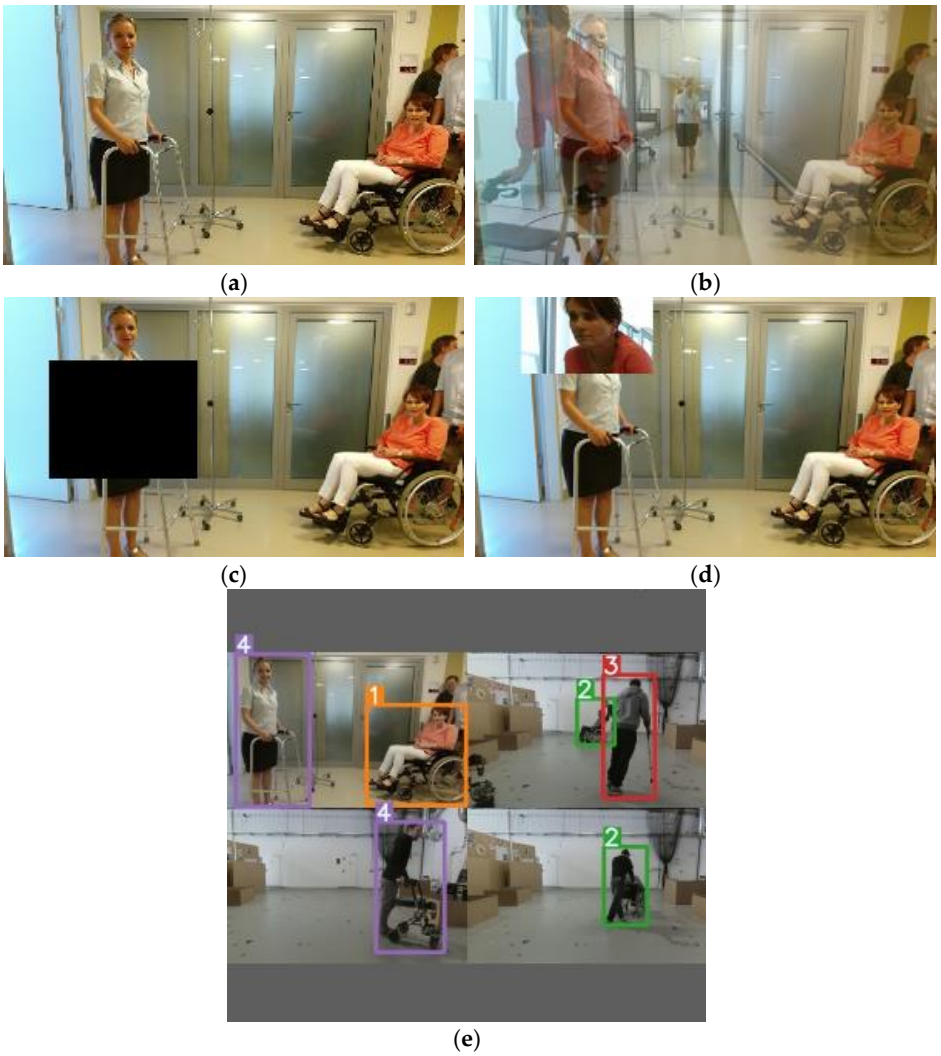


Figure 7. Examples of image augmentation. (**a**) Original. (**b**) MixUp 50%. (**c**) CutOut. (**d**) CutMix. (**e**) Mosaic.

4. Materials and Methods

Starting from a YOLOv4 implementation, this paper proposes a few modifications:

- AdamW as the new gradient descent optimizer;
- Addition of random noise to the images at runtime during training;
- A second stage regressor and classifier based on the Faster R-CNN classification and regression head;
- An anchor free module that is attached to every pyramid level, based on the paper written by Zhu et al. [60] and the implementation of [61];
- Removed MixUp and CutMix augmentations, as they led to poor performance.

The chosen architecture has three pyramid levels that are used for detecting people at different scales. For example, when the input image is of size 640 × 640, the output feature maps of the first pyramid level, which is called P3, will be of size 80 × 80, and the detection head will be responsible for finding people that are far away (smaller objects). The next two pyramid levels will further downscale the feature maps by a factor of 2 each, resulting in feature maps of size 40 × 40 on the second pyramid level (P4) and 20 × 20 on the third pyramid level (P5), with the last pyramid level being responsible for detecting people that are close to the camera (large objects). Due to the size of the images in the dataset and the average distance between the camera and the people that were recorded, the addition

of further pyramid levels is not needed as they would just increase the inference time of the model.

4.1. Dataset and Data Loader

Having chosen two datasets with very different annotation styles, either a separate data loader could be implemented for each of them, or both datasets could be converted to use the same annotation style. Since creating separate data loaders would have increased the risk of implementation errors, both datasets were converted to COCO annotation format.

The reason for not choosing to train and test on COCO testval dataset is the time required for the model to train. COCO train set contains around 118 k images, and the test set has around 5 k. In comparison, HospitalAids dataset, which is used, has around 11 k train images and 4 k test images and takes around 12 h to train for 80 epochs.

4.2. Gradient Descent Optimizer and Learning Rate Scheduler

Multiple gradient descent optimizers with different parameters were tested: Adam, AdamW, and AdamS [62]. Both Adam and AdamS led to poor performance on both datasets, creating many low-accuracy detections during the test phase. On the other hand, AdamW achieved good results, obtaining comparable results with Vasquez et al. [47,63] on the MobilityAids dataset and acceptable results on the MOT17Det dataset. More about these results can be found in Section 5.

As for the learning rate scheduler, there are multiple options available, the most common one being the step learning rate decay. However, this method requires fine-tuning the steps at which the learning rate decays, and the decay rate. To avoid these parameters, the cosine annealing learning rate scheduler proposed by Loshchilov et al. [53] was chosen; it only requires the starting learning rate, the final learning rate, and the number of epochs the algorithm will train for. This method changes the learning rate after each epoch, according to the equation:

$$\eta_t = \left(0.2 + \frac{(1.0 - 0.2)}{2} \cdot (1 + \cos(t \cdot \frac{\pi}{epochs}))\right) \cdot \eta_0, \quad (4)$$

which means that the learning rate will range between 100% to 20% of the original value in accord with a cosine function. Another advantage of cosine decay is that it decreases the learning rate after every epoch, with smaller changes at the beginning of the training process when the model should have a higher learning rate to search for minimum points, and at the end, when it fine-tunes its parameters and more drastic changes in between. In comparison, step scheduling makes more drastic changes when a certain number of steps have been reached. An example of how the learning rate varies can be seen in Figure 8.

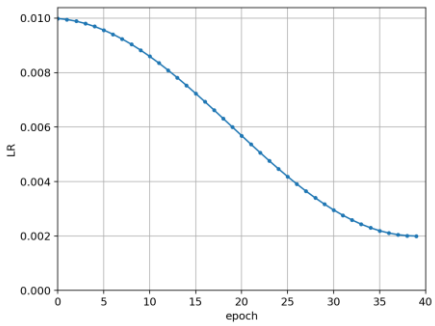


Figure 8. The plot created by the cosine annealing learning rate scheduler, starting from a value of 10^{-2} and decaying to a final value of 2×10^{-3} over the course of 40 epochs.

4.3. Dataset Augmentation Methods

Different dataset augmentation methods have been tested and the most efficient ones are:

- Left-right flips: The model becomes more robust to the different trajectories of people in the images;
- HSV alterations: The model becomes more robust to changes in illumination;
- ISO noise: An image could always be of lower quality because of the camera's sensor limitations;
- Mosaic: Training on multiple images at the same time increases diversity.

Up-down flips are unnatural for the people detection task, as it is highly unlikely that a person will be upside down in any test image. Rotations could prove useful but would introduce more hyperparameters that require fine-tuning. Scaling caused trouble when combined with Mosaic, as it would crop the images to match the original size, without removing the irrelevant annotations. CutMix and MixUp did not work well with the chosen datasets, as there was a very high chance of two images being similar. Figure 9 shows an example of a mixture of these data augmentation methods being applied on four different images from the MobilityAids dataset, such as Mosaic (top right and bottom left), ISO noise (top right), HSV alterations (either an increase or decrease in brightness/contrast in all images), LR flips (most images), as well as the now-removed scaling (top left and bottom right).

Figure 9. Samples from the training phase of a batch of 4 images from the MobilityAids dataset augmented with Mosaic, HSV alterations, ISO noise, and LR flips and scaling.

4.4. A New Classification and Regression Branch

This has been primarily added for regressing the new bounding boxes of the previously detected people in a sequence of images but can also be used as a second proposal head in the object detection task. The way it works is as follows:

1. The image is fed to the network composed of CSPDarknet53 and PANet in order to obtain the initial proposals, which we will denote as P.
2. The proposals P are then sent to three separate YOLO heads for two purposes:
   a. Extract the feature maps, F_m, and send each of them to the ROI pooling
   b. Obtain the proposed detections from YOLOv4 based on P and F_m, denoted as P_d
3. Send F_m and P_d to the ROI pooling layer and use the classification and regression head to obtain the new set of proposals P_s.
4. Send P_d and P_s to an NMS layer in order to filter any low-quality proposals and obtain our final set of proposals P_f.

The proposed architecture can be seen in Figure 10.

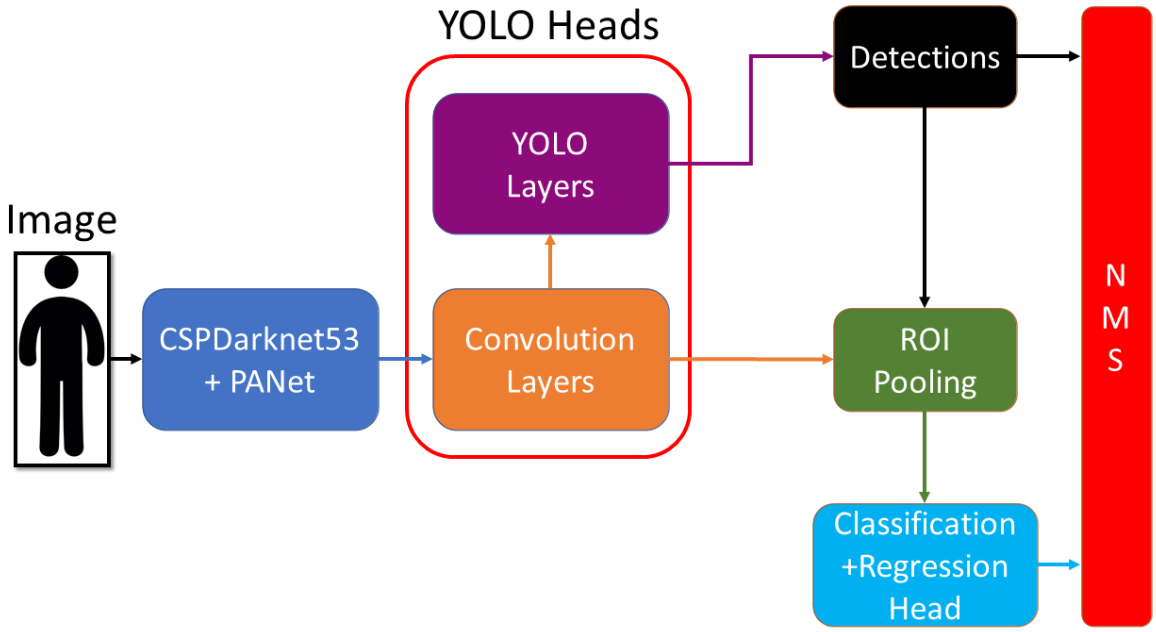


Figure 10. The proposed architecture of the system, based on 3 different YOLO Heads; each head contributes separately to the detection and ROI pooling part.

4.5. Anchor Free Branch

Because some ground truth boxes are either low-quality or are not covered by the anchor boxes, an anchor-free module attached to the existing architecture has been used. The method used is the one proposed by Zhu et al. [60]. The targets are processed for each pyramid level, creating a map of the same size as the feature maps which contains where the targets should be. The targets are further processed into effective (the area that is considered to contain the target object) and ignore (the area surrounding the effective zone and that is considered ambiguous and, as such, is ignored during backpropagation) areas that are a percentage of the main bounding box. Figure 11 shows what an example target would look like.

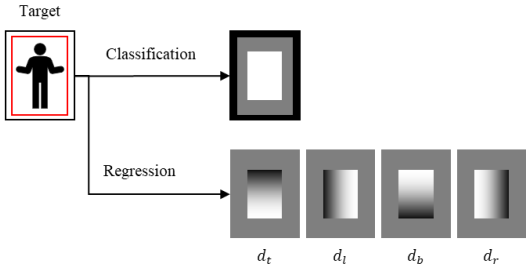


Figure 11. Classification and regression targets after being processed. The white rectangle inside the classification target is the effective area of the bounding box, the ignore area is colored in gray, and the gradients from this zone are not propagated, while the black area does not contain any object. The same goes for the regression target, but instead of one map, it has four (one for each distance between the current location and the location of the top, left, bottom and right sides of the original bounding box).

Because the module is attached to three different pyramid levels and predicting a low-quality bounding box is not desirable, each module only predicts the targets that best suit the pyramid level it is attached to. To do this, during training a target is assigned to the module whose output loss is the lowest during what the authors call Online Feature Selection.

Just as with the classification and regression head, this module is fed the feature maps of the layers before the YOLO head of each pyramid level.

4.6. Test Architecture and Coding Libraries

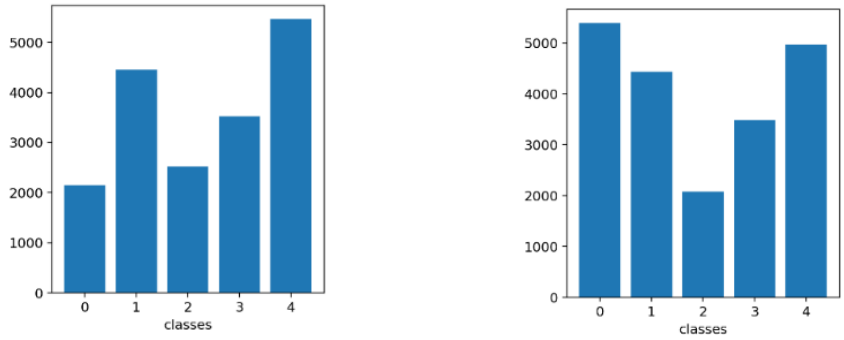
In order to train and test our network, we used the Python programming language, version 3.7, and the PyTorch library, version 1.7. Both the programming language and libraries were chosen for ease of use, as they are amongst the most common approaches in the field. All the validations were performed on a machine containing an Intel I9-9900k processor and an NVIDIA RTX2080 graphics card.

5. Results

The metrics used to measure the performance of an algorithm are mAP@0.5 and mAP@0.5:0.95 (see Appendix A.3). In addition to these, Precision and Recall are also used to better visualize the strengths and weaknesses of the model.

5.1. Results on the MobilityAids Dataset

On this dataset, there have been three notable experiments. By default, the dataset is split into a training set (10,934 images) and an evaluation set (4301 images), on which the authors of [63] have tested their algorithm. This split will be called Hospital1. However, from our observations, mAP and recall start to stagnate after 40 epochs, even after applying image augmentations during training, and fine-tuning the model. To test whether it was a problem of the dataset (insufficient data) or a problem of the train-evaluation split, a new, random split (11,415 images for training and 3820 images for evaluation) was created. Given that the dataset is composed of sequences of images, the splitting process had to take into consideration that a sequence must not contain images in both the train and validation sets. This split will be called Hospital2. The histograms of the number of images in which an object with class c appears can be seen in Figure 12.



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

Figure 12. Images per class distribution. The classes are: 0 = person without any handicap; 1 = person using crutches; 2 = person in a wheelchair; 3 = person in a wheelchair that is pushed by another person; 4 = person using a walking frame. (**a**) Hospital1 (**b**) Hospital2.

The correlation between the number of images that contain class 0 (a person without any handicap) and the performance of the algorithm (mAP) can be seen in Figures 12 and 13. However, to be able to compare the performance of this paper's approach to that of Vasquez et al. [63], the algorithm must be trained and tested on their split (Hospital1). To close the gap in performance between the two splits, the dataset could either be augmented with additional images that contain people of class 0 or use the weights of YOLOv4 pre-trained on COCO (as it contains such images and can help the algorithm further differentiate between objects), with the method of choice being the latter. This third experiment was called Pretrain_Hospital1. This experiment had a better starting accuracy than both previous experiments, required fewer epochs to reach the plateau, and had better accuracy overall than Hospital1, but had worse performance in the end than Hospital2, as can be seen in Figure 13. The performance of this approach compared to the methods proposed by Vasquez et al. [47,63] can be seen in Table 2. Compared to Vasquez et al. [47,63] methods, this approach does not use the depth information. Examples of output from this report's model can be found in Appendix B.

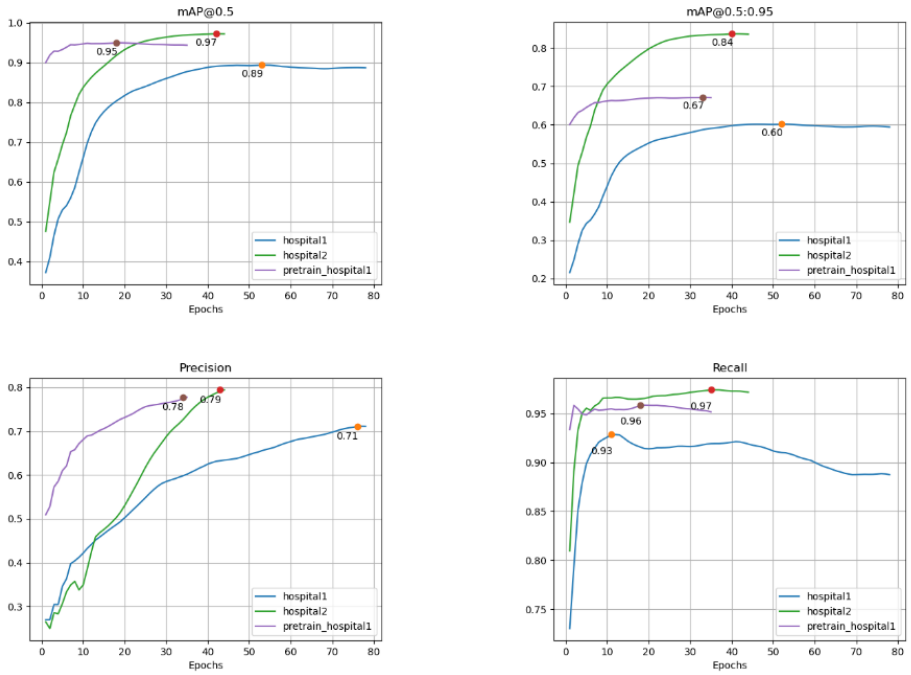


Figure 13. Performance metrics for the Mobility Aids dataset.

Table 2. Accuracy (mAP@0.5) comparison between this paper's method and those of [47,63].

Algorithm	Backbone	mAP@0.5	Inference Time (ms)
Fast R-CNN [47]	ResNet-50	64.98	43
Fast R-CNN [63]	ResNet-50	79.16	43
Faster R-CNN Centroid [63]	ResNet-50	96.52	110
Proposed Method	CSPDarknet53	95.00	22

5.2. Results on the MOT17Det Dataset

On this dataset there were 2 notable experiments:

- MOT_1, which uses the weights of YOLOv4 that were obtained after training the algorithm on the MobilityAids dataset (Pretrain_Hospital1);
- MOT_2, which uses the weight of YOLOv4 pre-trained on COCO dataset.

Since MOT17Det only provided ground truth data on the train set, the final train-test split was the following:

- train set: MOT17-02, MOT17-05, MOT17-10, MOT17-13 (2841 images)
- test set: MOT17-09, MOT17-11 (1425 images)

This split was chosen based on:

- the dynamic state of the camera (stationary or mobile):
  - stationary: MOT17-02, MOT17-04, MOT17-09;
  - mobile: MOT17-05, MOT17-10, MOT17-11, MOT17-13.
- the location of the scenes:
  - indoors: MOT17-09, MOT17-11;
  - outdoors: MOT17-02, MOT17-05, MOT17-10, MOT17-13.
- the average number of people in an image (the occluded people were removed from the annotations).

As such, the scenes recorded with a mobile camera in an outdoor environment were preferred to be used in the train set, as they offer more diversity in terms of background and illumination. Because of this, MOT17-09 was considered a good candidate for the test set. MOT17-11 was chosen for the test set due to the higher population density (compared to MOT17-09) and because it was recorded using a mobile platform. Out of the seven sequences of images, MOT17-04 had a different perspective (aerial view) and it was not used at all.

As can be seen in Figure 14, in both experiments the number of false positives decreased, while the number of false negatives increased as the model trained, which led to a decreasing mAP. This was caused by the small size of the dataset and the high instance of partially occluded people, which the algorithm had difficulty detecting (see Figure A4).

In terms of experiment comparison, MOT_2 had a higher maximum mAP than MOT_1 because in the MobilityAids dataset, on which the algorithm had been previously trained in the case of MOT_1, people were being differentiated based on their physical handicap, which was not relevant for this dataset.

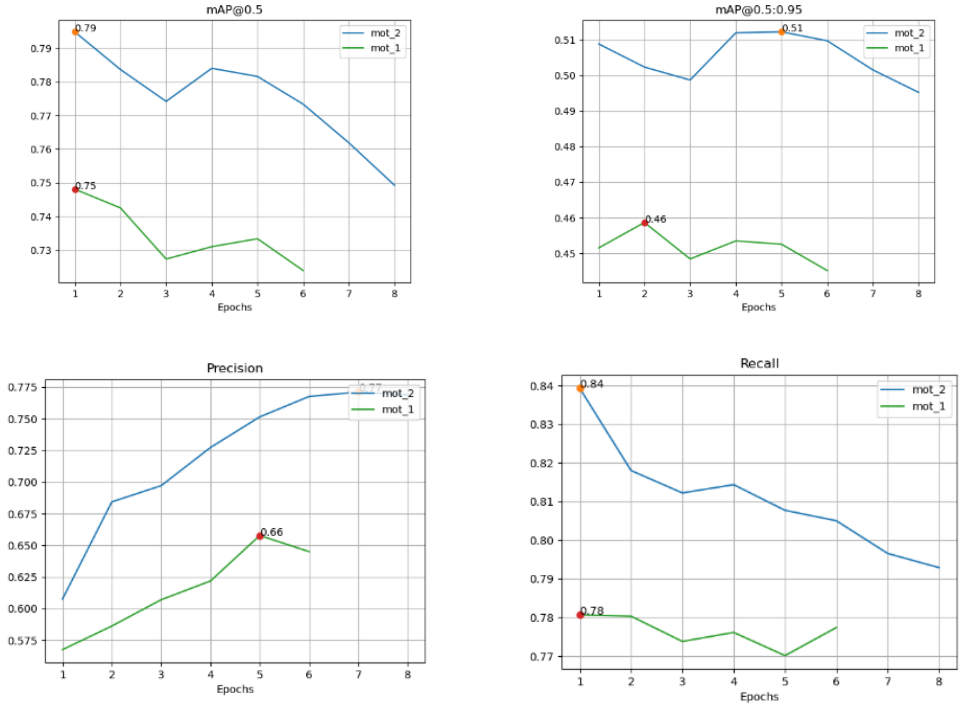


Figure 14. Performance metrics for the MOT17Det dataset.

6. Discussion

Two of the main factors when considering object detection are inference time and accuracy, and most work that is done in the domain requires some trade-off between the two. With this purpose in mind, most prefer the former over the latter, as the benefits gained from faster detection times tend to outweigh the detriments of lower overall accuracy. This is especially the case when considering people detection, and fast response in a critical situation tends to be the beneficial approach. This does not necessarily mean that a lower inference time must come with lower accuracy, as can be seen by developments in the field in recent years.

Papers such as [4–9] tried to improve upon the previous two-stage detector architecture, managing to improve both inference times and accuracy in a diverse range of applications. As for the case of single-stage detectors, the approaches have followed a similar direction, with each version of YOLO seeking to increase the performance by virtue of a smaller overall architecture [12–14,24].

Something else to note is that there have been various other papers that aimed at improving the overall architecture of YOLOv4 itself, with various different approaches [15,18,21,23].

Another thing of note is that both the resolution of the images used in the training set and the types of images used can largely affect the end results. Whilst a higher resolution image can increase the overall inference time, it also tends to produce more accurate results. Paper [64] shows a comparison between how different architectures perform in this regard. Variations in the images used in the training dataset tend to be beneficial, as can be seen in [14,58,59].

Even though most research tends to focus on more minimal architectures in order to improve overall inference times, our goal was to combine both the versatility of single-stage detectors with the accuracy offered by that of two-stage object detectors and aim for low inference times and high accuracy, which, for our dataset, we mostly achieved. One of the main downsides of our approach, however, was the fact that we could only find new bounding boxes where the YOLOv4 architecture had already identified them ahead of time, except they were given a low confidence score. This could improve the odds of some more obscure objects being detected in certain images; however, it also posed the issue that we

can run into more incorrect estimates, specifically since the box was given a low confidence score initially.

Another limitation of our model was the fact that it achieved poor results when trained on the MOT17Det dataset, most likely since many of the images were highly occluded; thus, our method was detrimental in this situation.

7. Conclusions and Future Directions

Starting from the architectures of Fast R-CNN and YOLOv4, this paper proposed a hybrid model of the two, using YOLOv4 as the main object detection algorithm and attaching the Fast R-CNN head to each pyramid level. The output of the Fast R-CNN head was then concatenated with that of YOLOv4, and non-maximum suppression was applied over them, gaining additional possible detections with almost no inference time difference compared to using just YOLOv4.

This method obtained comparable results with the methods proposed in the paper that introduced the MobilityAids dataset in terms of mAP@0.5, but at a lower inference time, as this was a single-stage object detector. On the MOT17Det dataset, on the other hand, even if the results were good, based on the metrics, there was no comparison with other existing methods, because the model was trained and tested locally on a split of the original train set. In the future, the model could be trained and tested on the entire dataset, but the results posting on the MOT_Challenge website are limited and, as can be seen in Section 5.2, the model has trouble learning on this dataset. Because of this, different training strategies must be employed for this dataset. A possibility of why the model is unable to learn is the high occlusion rate between people. Thus, as a future improvement, a method that would be able to still detect people in highly occluded images would be desired.

While not perfect, the idea of a regression and classification branch for a single-stage object detector might be improved upon. For example, in the case of Tracktor, the regression and classification branches are used to re-detect people from previous frames while using YOLOv4 for people detection in the current frame to reduce the inference time. Another possibility that was not tested would be bounding box confidence rescoring for the bounding boxes that were outputted by YOLOv4.

Author Contributions: Conceptualization, F.D. and C.-A.B.; Data curation, F.D. and M.-L.V.; Formal analysis, F.D. and M.-L.V.; Funding acquisition, C.-A.B.; Investigation, F.D. and M.-L.V.; Methodology, F.D. and C.-A.B.; Software, F.D.; Supervision, C.-A.B.; Validation, F.D. and M.-L.V.; Visualization, F.D., Costin-Anton Boiangiu and M.-L.V.; Writing—original draft, F.D.; Writing—review and editing, C.-A.B. and M.-L.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge Nicolae Tarbă for his great support and reviews.

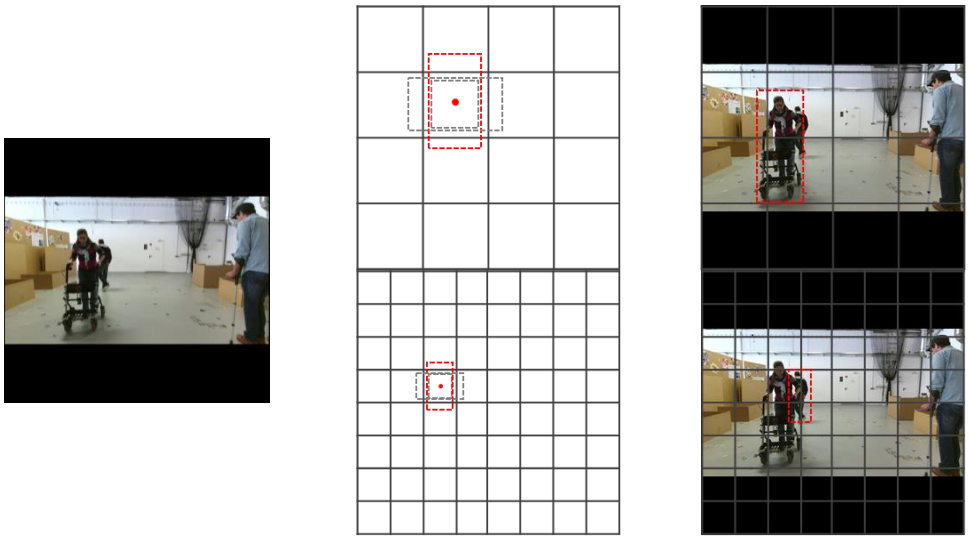
Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Anchor Boxes

Anchor boxes are predefined bounding boxes, which are characterized by width and height. In the case of Faster R-CNN, the anchor boxes are used on a sliding window to find candidate bounding boxes, while in YOLOv3 they are placed in each grid cell, with the center coordinates of the bounding box being aligned with the center of the grid cell or sliding window. A candidate bounding box is said to be found if it overlaps with a bounding box with an IoU greater than a threshold. Anchors are usually fine-tuned to a

certain dataset, as the objects in that dataset can have variable sizes and the anchor boxes must be able to generalize. YOLOv3 uses k-means to find the anchor boxes that best cover all ground-truth targets in the dataset, so they do not need to be hand-crafted. Starting from an anchor box, a bounding box is computed via offsetting the center of the anchor box and scaling it to fit the object of interest. An example of how they work in a single-stage object detector can be seen in Figure A2.



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

Figure A1. Example of how anchor boxes work at different scales. (**a**) Original padded image taken from the MobilityAids dataset. (**b**) Anchor boxes at two different scales. The 4×4 grid, at the top, captures bigger objects, while the 8×8 grid, at the bottom, captures finer details. (**c**) The anchor boxes are offset and scaled to fit the person at each scale.

Appendix A.2. Intersection over Union (IoU)

Intersection over Union (*IoU*) is a metric used to determine the similarity between two bounding boxes $b_1 = (x_1, y_1, w_1, h_1)$ and $b_2 = (x_2, y_2, w_2, h_2)$, in the task of object detection, and is defined as:

$$IoU = \frac{\text{Intersection area}}{\text{Union area}}, \tag{A1}$$

and can have values in the interval [0,1], where 0 means that the two bounding boxes do not intersect, while 1 indicates that they are identical. Figure A1 shows what each area represents.

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

Figure A2. Examples of different types of bounding box areas; (**a**) intersection (**b**) union.

Appendix A.3. Average Precision (AP)

An object detection algorithm has two tasks: (1) propose bounding boxes that might contain objects of interest and (2) identify the contained objects. To test the performance of the algorithm on a dataset, each prediction (sorted in decreasing order by the confidence score) is compared to the ground truth bounding boxes for each class and is labeled as:

- True Positive (*TP*) if:
  - The Intersection over Union (*IoU*) between the predicted bounding box and the ground truth is greater than or equal to a set threshold ($IoU \geq$ threshold) and the bounding box has been classified correctly
- False Positive (*FP*) if:
  - The Intersection over Union (*IoU*) between the predicted bounding box and the ground truth is below a set threshold ($IoU <$ threshold) and the bounding box has been classified correctly
  - A detection with a higher confidence score has already been assigned to the ground truth bounding box
- False Negative (*FN*) if:
  - No detection ($IoU = 0$)
  - The bounding box has been classified incorrectly

After each prediction labeling, the precision (*P*) and recall (*R*) over the accumulated statistics (*TP*, *FP*, *FN*) are computed, where:

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}, \tag{A2}$$

$$R = \frac{TP}{FP + FN} = \frac{TP}{\text{all ground truths}}, \tag{A3}$$

which are used to construct the precision-recall (PR) curve (for each recall value, find the maximum precision value).

The Average Precision (AP) metric for a single class is defined as the area below the PR curve at a set *IoU* threshold and as such, is usually written as AP@threshold, with the most common notations being AP@0.50 (AP at $IoU \geq 0.5$) and AP@0.5:0.95 (the average of AP values at 10 different *IoU* thresholds, ranging from 0.5 to 0.95 in 0.05 increments). mAP@threshold is the mean AP@threshold over all classes.

Appendix B

Figures A3 and A4 show examples of detection results obtained from different experiments on the MobilityAids dataset.

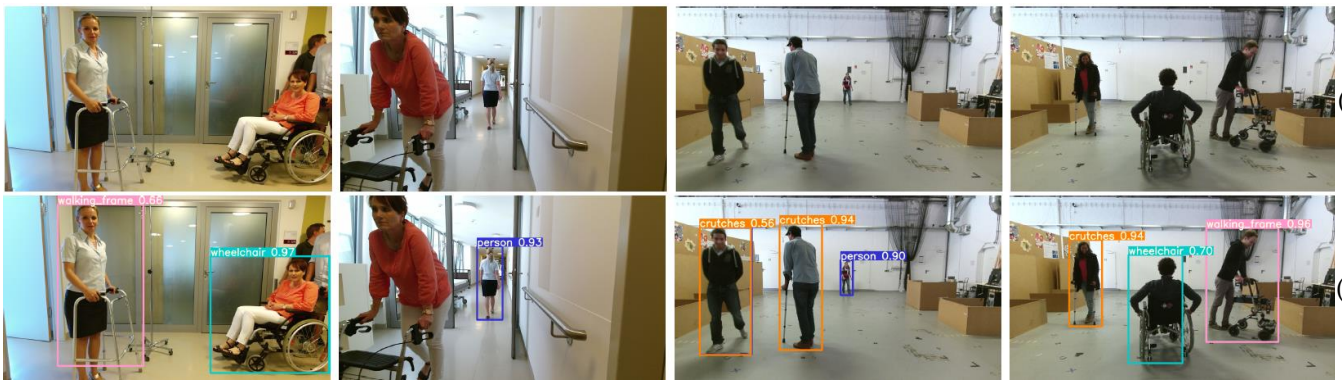


Figure A3. *Cont.*

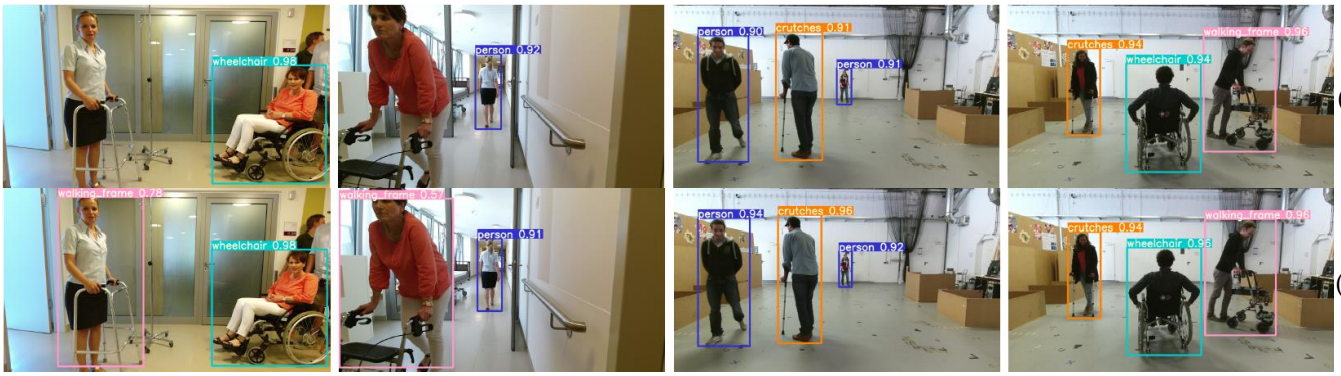


Figure A3. Detection results on 4 images from MobilityAids dataset. (**a**) Original image samples. (**b**) Experiment 1 (Hospital1) output sample. (**c**) Experiment 2 (Hospital2) output sample. (**d**) Experiment 3 (Pretrained_Hospital1) output sample.

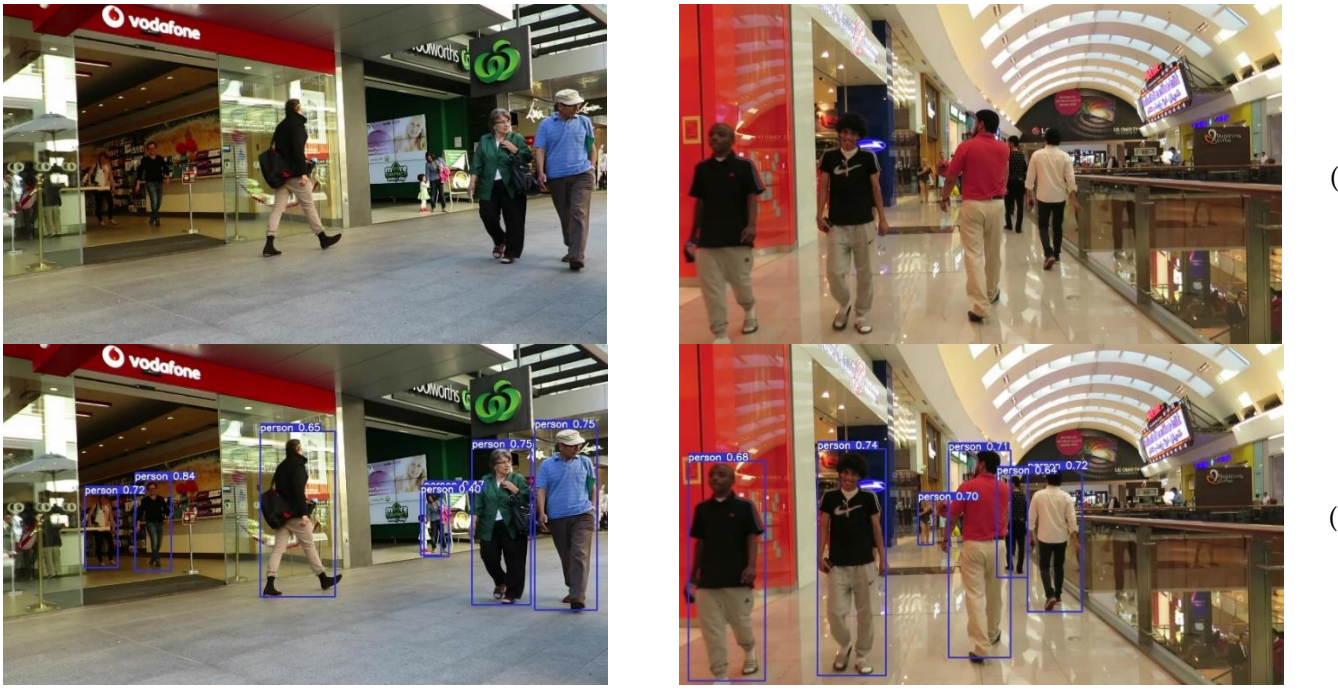


Figure A4. Detection results on 2 images from the MOT17Det dataset. (**a**) Original image samples. (**b**) Output samples.

References

1. Viola, P.; Jones, M. Rapid Object Detection Using a Boosted Cascade of Simple Features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–16 December 2001; Volume 1, p. I. [CrossRef]
2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013. [CrossRef]
3. 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; pp. 770–778. [CrossRef]
4. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [CrossRef]
5. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
6. Li, L.; Ma, J. Zenithal People Detection Based on Improved Faster R-CNN. In Proceedings of the 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Chengdu, China, 7–10 December 2018; pp. 1503–1508. [CrossRef]

7. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving Into High Quality Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6154–6162. [CrossRef]
8. Ren, X.; Du, S.; Zheng, Y. Parallel RCNN: A Deep Learning Method for People Detection Using RGB-D Images. In Proceedings of the 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 14–16 October 2017; pp. 1–6. [CrossRef]
9. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988. [CrossRef]
10. Wang, Q.; Li, W.; Liu, H.; Shan, L. A Robust Approach for Students Detection via Multi Cameras with Mask-RCNN. In Proceedings of the 2021 2nd International Conference on Computers, Information Processing and Advanced Education CIPAE, Ottawa, ON, Canada, 25–27 May 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 24–28. [CrossRef]
11. 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. [CrossRef]
12. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. [CrossRef]
13. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
14. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
15. Wang, H. ProYOLOv4: Some Improvements for YOLOv4. In Proceedings of the 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), Beijing, China, 14 May 2021; pp. 879–883. [CrossRef]
16. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
17. Ahmad, M.; Ahmed, I.; Ullah, K.; Ahmad, M. A Deep Neural Network Approach for Top View People Detection and Counting. In Proceedings of the 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), New York, NY, USA, 10–12 October 2019; pp. 1082–1088. [CrossRef]
18. Zhang, Z.; Xia, S.; Cai, Y.; Yang, C.; Zeng, S. A Soft-YoloV4 for High-Performance Head Detection and Counting. *Mathematics* **2021**, *9*, 3096. [CrossRef]
19. Peker, M.; İnci, B.; Musaoğlu, E.; Çobanoğlu, H.; Kocakır, N.; Karademir, Ö. An Efficient Deep Learning Framework for People Detection in Overhead Images. In *Artificial Intelligence in Industrial Applications: Approaches to Solve the Intrinsic Industrial Optimization Problems*; Fernandes, S.L., Sharma, T.K., Eds.; Learning and Analytics in Intelligent Systems; Springer International Publishing: Cham, Switzerland, 2022; pp. 1–20. [CrossRef]
20. Sambolek, S.; Ivasic-Kos, M. Automatic Person Detection in Search and Rescue Operations Using Deep CNN Detectors. *IEEE Access* **2021**, *9*, 37905–37922. [CrossRef]
21. Wen, W.; Xia, F.; Xia, L. Real-Time Personnel Counting of Indoor Area Division Based on Improved YOLOV4-Tiny. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 10–23 October 2021; pp. 1–6. [CrossRef]
22. Yu, H.; Chen, W. Motion Target Detection and Recognition Based on YOLOv4 Algorithm. *J. Phys. Conf. Ser.* **2021**, *2025*, 012053. [CrossRef]
23. Kumar, A.; Kalia, A.; Sharma, A.; Kaushal, M. A Hybrid Tiny YOLO V4-SPP Module Based Improved Face Mask Detection Vision System. *J. Ambient. Intell. Hum. Comput.* **2021**, 1–14. [CrossRef]
24. Long, X.; Deng, K.; Wang, G.; Zhang, Y.; Dang, Q.; Gao, Y.; Shen, H.; Ren, J.; Han, S.; Ding, E.; et al. PP-YOLO: An Effective and Efficient Implementation of Object Detector. *arXiv* **2020**, arXiv:2007.12099.
25. Sun, J.; Ge, H.; Zhang, Z. AS-YOLO: An Improved YOLOv4 Based on Attention Mechanism and SqueezeNet for Person Detection. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 March 2021; Volume 5, pp. 1451–1456. [CrossRef]
26. Kusuma, T.A.A.H.; Usman, K.; Saidah, S. People Counting for Public Transportations Using You Only Look Once Method. *J. Tek. Inform. (Jutif)* **2021**, *2*, 57–66. [CrossRef]
27. Abdusalomov, A.; Mukhiddinov, M.; Djuraev, O.; Khamdamov, U.; Whangbo, T.K. Automatic Salient Object Extraction Based on Locally Adaptive Thresholding to Generate Tactile Graphics. *Appl. Sci.* **2020**, *10*, 3350. [CrossRef]
28. Sharma, V.; Mir, R.N. A Comprehensive and Systematic Look up into Deep Learning Based Object Detection Techniques: A Review. *Comput. Sci. Rev.* **2020**, *38*, 100301. [CrossRef]
29. Sultana, F.; Sufian, A.; Dutta, P. A Review of Object Detection Models Based on Convolutional Neural Network. In *Intelligent Computing: Image Processing Based Applications*; Mandal, J.K., Banerjee, S., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2020; pp. 1–16. [CrossRef]
30. Dhillon, A.; Verma, G.K. Convolutional Neural Network: A Review of Models, Methodologies and Applications to Object Detection. *Prog. Artif. Intell.* **2020**, *9*, 85–112. [CrossRef]
31. Oksuz, K.; Cam, B.C.; Kalkan, S.; Akbas, E. Imbalance Problems in Object Detection: A Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3388–3415. [CrossRef]

32. Zhao, Z.-Q.; Zheng, P.; Xu, S.-T.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]
33. Tong, K.; Wu, Y.; Zhou, F. Recent Advances in Small Object Detection Based on Deep Learning: A Review. *Image Vis. Comput.* **2020**, *97*, 103910. [CrossRef]
34. Bergmann, P.; Meinhardt, T.; Leal-Taixé, L. Tracking Without Bells and Whistles. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 941–951. [CrossRef]
35. Schapire, R.E. A Brief Introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*; IJCAI'99; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1999; Volume 2, pp. 1401–1406.
36. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893. [CrossRef]
37. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support Vector Machines. *IEEE Intell. Syst. Appl.* **1998**, *13*, 18–28. [CrossRef]
38. Uijlings, J.R.R.; van de Sande, K.E.A.; Gevers, T.; Smeulders, A.W.M. Selective Search for Object Recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
39. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021; pp. 13024–13033. [CrossRef]
40. Papers With Code. Available online: https://paperswithcode.com/sota/object-detection-on-coco (accessed on 2 May 2021).
41. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 22–25 July 2017; pp. 936–944. [CrossRef]
42. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef]
43. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: A Simple and Strong Anchor-Free Object Detector. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, 1. [CrossRef]
44. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. *Int. J. Comput. Vis.* **2020**, *128*, 642–656. [CrossRef]
45. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27–28 October 2019; pp. 6568–6577. [CrossRef]
46. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as Points. *arXiv* **2019**, arXiv:1904.07850.
47. Vasquez, A.; Kollmitz, M.; Eitel, A.; Burgard, W. Deep Detection of People and their Mobility Aids for a Hospital Robot. In *2017 European Conference on Mobile Robots (ECMR)*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7.
48. Dendorfer, P.; Ošep, A.; Milan, A.; Schindler, K.; Cremers, D.; Reid, I.; Roth, S.; Leal-Taixé, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. *Int. J. Comput. Vis.* **2021**, *129*, 845–881. [CrossRef]
49. MOT Challenge—Data. Available online: https://motchallenge.net/data/MOT17Det/ (accessed on 22 January 2021).
50. Milan, A.; Leal-Taixe, L.; Reid, I.; Roth, S.; Schindler, K. MOT16: A Benchmark for Multi-Object Tracking. *arXiv* **2016**, arXiv:1603.00831.
51. COCO—Common Objects in Context. Available online: https://cocodataset.org/#detection-2020 (accessed on 22 January 2021).
52. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
53. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. *arXiv* **2019**, arXiv:1711.05101.
54. 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 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 658–666. [CrossRef]
55. Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; Yang, Y. Random Erasing Data Augmentation. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 13001–13008. [CrossRef]
56. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
57. DeVries, T.; Taylor, G.W. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv* **2017**, arXiv:1708.04552.
58. Yun, S.; Han, D.; Chun, S.; Oh, S.J.; Yoo, Y.; Choe, J. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27–28 October 2019; pp. 6022–6031. [CrossRef]
59. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. *arXiv* **2018**, arXiv:1710.09412.
60. Zhu, C.; He, Y.; Savvides, M. Feature Selective Anchor-Free Module for Single-Shot Object Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 840–849. [CrossRef]
61. FSAF Implementation. Available online: https://github.com/hdjang/Feature-Selective-Anchor-Free-Module-for-Single-Shot-Object-Detection (accessed on 21 January 2021).
62. Xie, Z.; Sato, I.; Sugiyama, M. Understanding and Scheduling Weight Decay. *arXiv* **2021**, arXiv:2011.11152.

63. Kollmitz, M.; Eitel, A.; Vasquez, A.; Burgard, W. Deep 3D Perception of People and Their Mobility Aids. *Robot. Auton. Syst.* **2019**, *114*, 29–40. [CrossRef]
64. Carranza-García, M.; Torres-Mateo, J.; Lara-Benítez, P.; García-Gutiérrez, J. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sens.* **2021**, *13*, 89. [CrossRef]