Engineering Science and Technology, an International Journal

Evaluating Yolo Models for Outdoor Object Detection in Advanced Driver Assistance Systems

--Manuscript Draft--

Manuscript Number:	JESTCH-D-23-00121
Article Type:	Review Article
Keywords:	advanced driver assistance system; yolo; outdoor object detection; convolutional neural network; real-time.
Abstract:	Recent advances in neural networks models have boosted the state-of-the-art performance for many applications. In particular, computer vision application has the most of attention and many advances were achieved. Object detection models were developed based on convolutional neural networks for general object detection and through the transfer learning technique, they were reoriented for custom object detection. In the context of detecting objects by advanced driver assistance system (ADAS), most of the existing object detection models can be used with minor modifications. You look only once (yolo) model have many versions and all of them has achieved the state-of-the-art at a certain time. It was a model designed for real-time processing without degrading the precision. In this work, we propose to evaluate the performance of the different versions of the yolo model for outdoor object detection in an urban environment. The transfer learning technique was applied to evaluate the suitability of the yolo models for the detection of outdoor objects related to urban spaces. The BDD100K dataset was used to train and evaluate the yolo models. The achieved results were very effective. The yolo v1 has the fastest processing speed, the yolo v5 achieved the best accuracy and the yolo v3 model has the best trade-off between speed and precision. The presented evaluation has proved that many models are suitable for implementation for ADAS while others are not. Models with a good balance between precision, speed and model size are the best.

Abstract:

Recent advances in neural networks models have boosted the state-of-the-art performance for many applications. In particular, computer vision application has the most of attention and many advances were achieved. Object detection models were developed based on convolutional neural networks for general object detection and through the transfer learning technique, they were reoriented for custom object detection. In the context of detecting objects by advanced driver assistance system (ADAS), most of the existing object detection models can be used with minor modifications. You look only once (yolo) model have many versions and all of them has achieved the state-of-the-art at a certain time. It was a model designed for real-time processing without degrading the precision. In this work, we propose to evaluate the performance of the different versions of the yolo model for outdoor object detection in an urban environment. The transfer learning technique was applied to evaluate the suitability of the yolo models for the detection of outdoor objects related to urban spaces. The BDD100K dataset was used to train and evaluate the yolo models. The achieved results were very effective. The yolo v1 has the fastest processing speed, the yolo v5 achieved the best accuracy and the yolo v3 model has the best trade-off between speed and precision. The presented evaluation has proved that many models are suitable for implementation for ADAS while others are not. Models with a good balance between precision, speed and model size are the best.

Keywords: advanced driver assistance system, yolo, outdoor object detection, convolutional neural network, real-time.

1. Introduction

Recently, a huge advance in the computer vision field was achieved thanks to the use of convolutional neural network (CNN) models [1]. CNN models are the most used for computer vision applications due to their ability to learn autonomously for the input data and can process data and generate predictions in a way similar to the biological brain. It processes visual data through a big number of neurons where each neuron is responsible for a small receptive field then those small receptive fields are correlated together to form the entire image. Those advances make the CNN very effective for visual data processing. CNN models were successfully deployed for many computer vision applications such as object classification [2], [3], object detection [4], image segmentation [5], scene recognition [6], indoor object detection [7].

The recent advances in the computer vision field are paving the research for building intelligent transportation systems such as advanced driver assistance systems (ADAS) and autonomous vehicles. Generally, intelligent transportation systems are based on a big number of sensors such as lidar, radar, and cameras. Visual data is the richest data in information and can be used for many applications. Since CNN models are very effective in processing visual data it is considered the best solution for developing an intelligent transportation system.

CNN models have been widely used for intelligent transportation applications such as traffic light detection [8], traffic signs recognition and detection [9], [10], pedestrian detection [11], [12], and many other applications. Most of the used CNN models were designed for general purposes and then reused for intelligent transportation-related applications. The transfer learning technique allows the reuse of CNN models to solve new problems without major modifying in the architecture. Besides, this technique allows to speed up the training and grantee the convergence of the model to achieve good performance.

Outdoor object detection in an urban space is a fundamental asset because of its aim of building a safer environment for both drivers and pedestrians. The main idea is to spot and interrupt surrounding objects to warn the driver in dangerous situations and to enforce traffic rules respect. Objects in urban spaces have

different shapes and sizes. For example, there is a big difference between a traffic light, cars, and pedestrians. The fact that objects have a wide difference in color, shape, and size makes their detection and recognition a hard task. Nevertheless, building a robust outdoor object detection system is a constrained problem because of the real-world variations such as occlusion, differences in point of view, weather conditions (fog, rain, and dust), and luminosity (day, night, cloudy, sunny). The outdoor object detection system must take into account all those factors with a focus on real-time processing, which is crucial for ADAS.

In this work, an ADAS with an intelligent vision system was developed using an object detection model. The yolo object detection models [13] were evaluated for detecting objects in urban spaces. Until now, 5 versions of the yolo were proposed and each of them has its configuration. The first version of yolo was designed for speed purposes with low accuracy. The second version Yolo 9000 [14] was designed to enhance the accuracy without hurting the speed. Both models have a common problems that are struggle with detecting small and close objects. The third version yolo v3 [15] was proposed to find a balance between speed and accuracy. The achieved results were very good. The fourth and fifth versions, Yolo v4 [16] and Yolo v5 [17], were designed to enhance the performance.

The yolo models were originally trained on large-scale datasets such pascal voc [18], MSCOCO [19], and ImageNet [20]. Those datasets contain a huge number of images with annotated objects such as cars, cats, plants, people, and many other objects. The mean average precision (mAP) was considered as an effective evaluation metric in terms of precision for the combination of classification and regression tasks. since the yolo models have converged on those datasets and achieved good performance in detecting objects, they will be a good candidate for outdoor object detection in ADAS.

The yolo models were adapted for detecting objects in an urban environment through the use of the transfer learning technique. The performance of those models was evaluated and the best candidate was selected for the task. As evaluation metrics, the achieved mAP, the memory allocation, the number of floating-point operations (FLOPS), the number of parameters, and the processing time. Also, the effect of the input image size was discussed. In order to select the best model for the desired task, all the evaluation metrics were compared with a focus on finding a good trade-off between speed and accuracy. For an ADAS, real-time processing is a crucial parameter that must be achieved while having a good detection precision. Also, memory consumption and computation complexity are important parameters that allow the embedded implementation of the detection system.

In this paper, the 5 versions of yolo were evaluated for outdoor object detection in an urban environment where objects related to ADAS were considered. All the models were trained on the MS COCO dataset and fine-tuned on the BDD100K dataset [48] through the transfer learning technique. This dataset presents images captured using a camera mounted on a car roof with custom object annotations. The BDD100K dataset [48] was designed for many tasks such as lane detection, image segmentation, and road objects detection. In this work, our focus was on the road objects detection for ADAS.

To the best of our knowledge, there is no existing paper that evaluates the performance of all the yolo models for detecting road objects. Besides, multiple evaluation metrics such as mAP, memory consumption, number of FLOPs, and the number of parameters were considered to select the best fit model for the studied task. The main contributions of this paper are as follows: (1) presenting a complete and detailed explanation of the background of the yolo models and the main difference between the versions. (2) analyses and evaluate the performance of the yolo models for outdoor object detection for ADAS. The evaluation was based on multiple evaluation metrics to decide which version fits the best for the task. (3) compare the achieved results and discuss their relevance in order to judge the models for a new task and evaluate the

efficacity of the transfer learning technique. (4) selecting the best performance model based on analyzing different evaluation metrics with a focus on real-time processing, high performance, and small model size.

The rest of the paper is organized as follows: Section 2 presents related works on outdoor object detection in the urban environment. The background of the yolo models was detailed and explained in section 3. Experiments and results were presented in section 4. In section 5, the paper was concluded and future works were proposed.

2. Related works

Outdoor object detection is considered an important research field due to its importance for many computer vision applications such as surveillance and intelligent transportation. Building a reliable detection system in the outdoor scene is a very hard challenge because of real-world conditions but many works were proposed to boost state-of-the-art in this field.

Ning et al. [21] propose an outdoor object detection model based on the single-shot multi-box detector (SSD) [22] for the unmanned aerial vehicular system. The SSD was modified by adding an inception module in the extra layers of the model. The aim of this modification was the enhancement of the accuracy without degrading the processing speed. The proposed inception module was derived from the inception model [23] with some modifications. A residual connection and batch normalization layers were added. Besides, the non-maximum suppression algorithm was improved for better performance. The proposed model was evaluated on the pascal voc 2007 dataset [18] and an mAP of 78.6% was achieved. Also, the proposed model was evaluated on a custom-made dataset called OOD [21] and 79.9% of mAP was achieved.

An outdoor object detection method, FPSSD, for blind people was proposed in [24] based on the SSD model [22] with minor modification. A feature pyramid was added to the original SSD model to detect the variation in object sizes. In addition, a custom dataset was built for the desired task called BLIND. The FPSSD achieved an mAP of 75.4% on the BLIND dataset. the proposed FPSSD has an improvement of 1.7% compared to the original SSD model.

Wang et al. [25] proposed a pedestrian detection method in outdoor surveillance scenes. The proposed method was based on a CNN model with a focus on pedestrian positions indicated by hierarchical unsupervised guidance information, including motion information and static information. The proposed CNN combines the region-based CNN model (R-CNN) [26] with a spatial attention module, feature pyramid network [27], and a cascading module to generate the final output. The R-CNN backbone takes the RGB as input and the spatial module takes the motion information as input. The output feature maps of the R-CNN backbone and the spatial module were concatenated and used as input of the feature pyramid network. Then, each pyramid level output was processed through a region of interest pooling layer and all feature maps were cascaded to generate the final output of the model. The main finding of this work that they prove the importance of motion information to detect moving objects in an outdoor scene. An average precision (AP) of 59.1% was achieved on the DukeMTMC dataset [28]. Also, the proposed model was evaluated on a proposed dataset named DML dataset and an AP of 42.1% was achieved.

Work in [29] proposed a comparison between the yolo v3 model [15] and the SSD model [22] for advertising panel detection in the outdoor scene. To evaluate the performance of the models a custom dataset was built and annotated. The proposed dataset was in streets and from the internet resulting in a total of 1800 images. an additional 261 images were collected for testing purposes. To increase the number of images in the training process a data augmentation technique was applied and 5884 images was used for the train where 484 images were used for validation with the SSD model and 589 images used the validation

with the yolo model. The evaluation of both models on the collected dataset proved that the SSD model eliminates almost all false positive prediction and the yolo model achieves high localization and classification results and detects more true positive cases than SSD does.

Algredo-Badillo et al. [30] proposed an obstacle detection model in an outdoor scene for road safety. The proposed model was based on a lightweight CNN model for implementation on field-programmable arrays (FPGA). The proposed model merged different types of images from different sensors such as a Kinect sensor, an RGB camera, and an infrared camera. Information from different images was gathered and processed for better performance in different illumination conditions. The reported result has shown impressive results for CNN implementation on hardware low-power devices.

An outdoor scene classification was performed in [31] based on combining the yolo v2 model [14] with a semantic rule. First, the yolo v2 model was used to detect and identify the object in outdoor scenes. seven outdoor objects were considered in this work, which are sky, green_land, house, tree, sand_land, water, and windmill. Second, the semantic rule was used to classify the outdoor scene based on the detected objects. Six scene classes were identified, which are barn, beach, desert, windfarm, green_ground, and water_land. The yolo v2 model has achieved 79.35% of mAP and the entire classification approach has achieved an F-score of 76.81% on the SUN397 dataset. The achieved results proved that the performance of the outdoor scene classification approach relies on the performance of the object detection model.

3. Background of the yolo models

Yolo was the first object detection model based on a CNN that need to have a single pass through the input image to identify which of a known set of objects might be present and provide information about their positions. The main purpose of the yolo model was to achieve real-time processing with acceptable detection precision.

3.1. Yolo v1

In the first version of yolo, the identification and localization tasks were solved as a regression problem. The CNN backbone was used to process the input image and generates feature maps with a predefined resolution. The output feature maps were considered as a grid with $s \times s$ dimension. For the pascal voc dataset, yolo generates *7x7* grid and for each grid cell, only one object can be predicted. To detect objects, in a grid cell (n, n), red box in figure 1, a fixed number of bounding boxes (b), yellow boxes in the figure, were predicted. For the pascal voc dataset, 2 bounding boxes were predicted for each grid cell. Each predicted bounding box has 5 parameters, which are the (x, y) coordinate, the height (h) and the width (w) of the bounding box, and a confidence score (c) that reflects how likely the box contains an object. The values of the parameters (x, y, h, w) were normalized by the high and the width of the input image. So, all values (x, y, h, w) were normalized to be between zero and one. Also, a conditional probability for each class of the dataset (20 classes for pascal voc) was predicted. This probability defines the percentage that the detected object belongs to a particular class. So, the prediction tensor of the yolo model has a shape of (s, s, $bx5+c$).

The CNN backbone of yolo must generate an output tensor with a shape of (*7,7,30*). The yolo propose a CNN model that takes an input image with a shape *448x448x3*, the 3 denotes the RGB space color, and generate an output tensor with a shape of *7x7* with 1024 channels. The proposed CNN backbone was composed of 19 convolution layers followed each by a non-linear activation layer, 4 max-pooling layers

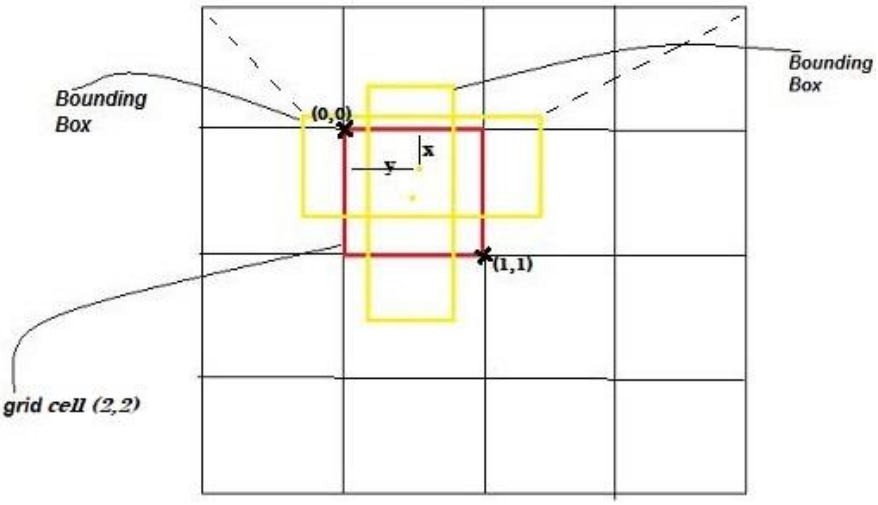

Figure 1: bounding boxes prediction by yolo

and 2 fully connected layers. For the first convolution layer, a 7x7 kernel size with a stride of 2 was used to have a quick look at the input image. Then for other convolution layers, 3x3 and 1x1 kernel sizes with a stride of 1were used except for a layer at the top of the network a stride of 2 was used. The 1x1 convolution layers were used to reduce the features space from preceding layers. For max-pooling layers, 2x2 kernel size with a stride of 2 was used. The architecture of the yolo model is illustrated in figure 2.

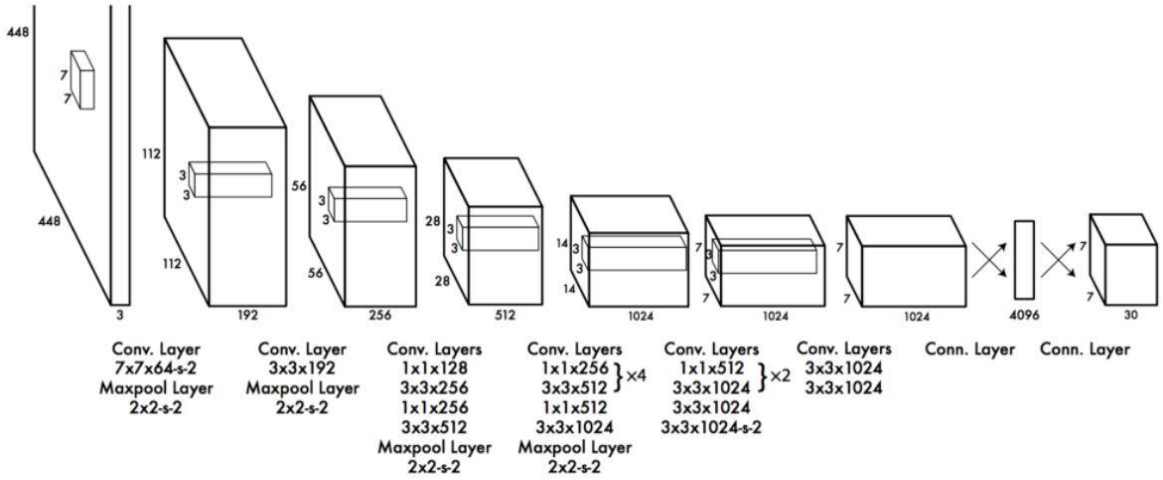

Figure 2: Architecture of the yolo model

To generate the final prediction, yolo keeps bounding boxes with the highest confidence score and eliminate other predictions. The class confidence score can be calculated through the multiplication between the bounding box confidence score and the conditional probability of the class. That can be used to measure the confidence for both identification and localization tasks simultaneously.

The final perdition bounding box must fit the detected object which has the highest intersection over union (IoU) value. To train the model for specializing the bounding boxes for different sizes and aspect ratios, the highest IoU value was selected to compute the loss function. Yolo loss function is composed of 3 sub-loss functions, classification loss, localization loss, and the confidence score. The loss function is the sum-square

error between the predictions and targets. The classification loss for each detected object is the square error of the class conditional probabilities. The classification loss can be computed as 1.

$$\textit{Classification loss} = \sum_{i=0}^{s^2} \mathbb{L}_i^{obj} \sum_{c \in classes} (p_i(c) - \widehat{p_i}(c))^2 \quad (1)$$

If an object is detected in cell *i*, $\mathbb{L}_i^{obj}$ takes 1 if no object is detected it takes 0. $p_i(c)$ is the target conditional class probability of class *c* in cell *i* and $\widehat{p_i}(c)$ is the predicted conditional class probability. S is the grid size. the localization loss was used to compute the error between the predicted bounding box and the ground truth bounding box. To accelerate the optimization only bounding boxes with detected objects were considered. To eliminate weighing absolute error in large bounding boxes and small bounding boxes equally the root square of the height and width were used instead of height and width. The localization loss is computed as 2.

$$localisation\ loss = \vartheta_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{L}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \vartheta_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{L}_{ij}^{obj} \left[(\sqrt{w}_i - \sqrt{\widehat{w}_i})^2 + (\sqrt{h}_i - \sqrt{\hat{h}_i})^2 \right] \quad (2)$$

Where $\mathbb{L}_{ij}^{obj}$ = 1 if a bounding box *j* is responsible for detecting an object in the cell *i*, otherwise it takes 0. (x_i, y_i, w_i, h_i) are the target bounding box coordinates and $(\hat{x}_i, \hat{y}_i, \widehat{w}_i, \hat{h}_i)$ are the predicted bounding box coordinates. The ϑ_{coord} was used to increase the weights for the loss of the bounding box coordinates. In yolo, the default value of ϑ_{coord} is 5 which puts more emphasis on the bounding box accuracy. If an object was detected in a bounding box, the objectness of this object must be measured. Also, if no object was detected in the bounding box, the objectness must be measured too to focus on false-positive samples. The confidence loss was used for this purpose. The confidence loss is computed as 3.

$$object\ confidence\ loss = \sum_{i=0}^{s^2} \sum_{j=0} \mathbb{L}_{ij}^{obj} (c_i - \widehat{c_i})^2$$

$$no\ object\ confidence\ loss = \vartheta_{noobj} \sum_{i=0}^{s^2} \sum_{j=0} \mathbb{L}_{ij}^{noobj} (c_i - \widehat{c_i})^2 \quad (3)$$

The yolo loss function combines the classification loss, the localization loss and the confidence loss. The loss can be computed as 4.

$$yolo\ loss = \vartheta_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{L}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \vartheta_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{L}_{ij}^{obj} \left[(\sqrt{w}_i - \sqrt{\widehat{w}_i})^2 + (\sqrt{h}_i - \sqrt{\hat{h}_i})^2 \right] + \sum_{i=0}^{s^2} \sum_{j=0} \mathbb{L}_{ij}^{obj} (c_i - \widehat{c_i})^2 + \vartheta_{noobj} \sum_{i=0}^{s^2} \sum_{j=0} \mathbb{L}_{ij}^{noobj} (c_i - \widehat{c_i})^2 + \sum_{i=0}^{s^2} \mathbb{L}_i^{obj} \sum_{c \in classes} (p_i(c) - \widehat{p_i}(c))^2 \quad (4)$$

As a final prediction yolo can generate a duplicated prediction for the same object. The non-maximum suppression (NMS) was used to eliminate redundant prediction by eliminating low confidence predictions. the NMS works as following: (1) sort all predictions based on the confidence score. (2) starting by the highest confidence score and eliminate any current predictions if any previous prediction for the same class

with an IoU lower than 0.5 has been found for the current prediction. (3) repeating the second step until all predictions have been checked. The NMS has added 2-3% to the mAP.

3.2. Yolo v2

Yolo v1 has demonstrated its performance but presents many limitations. It has high localization error and low recall, which measure the performance of locating all objects, and cannot detect close and small objects. The second version of yolo was designed to overcome the limitations of the yolo v1 by improving the detection performance and being faster. First, Yolo v2 started by modifying the backbone network. The new backbone has a residual connection at the top convolution layers, the last pooling layer was removed and for detection, fully connected layers were replaced with a convolution layer with a kernel size of 1x1. The input image has a resolution of 416x416 instead of 448x448 for the detection. The idea behind using this resolution was to achieve a grid size of 13x13 after 32x downsampling operations. A new training process was proposed by training the backbone for classification on the ImageNet using input images with a resolution of 224x224 then it was fine-tuned using input images with a resolution of 448x448. Those modification has enhanced the mAP by 4%. Second, a batch normalization layer was added after each convolution layer. Adding this optimization eliminates the need for other optimizations such as dropout without facing an overfitting problem. The batch normalization speeds up the training process and increases the convergence probability. This modification has boosted the mAP by 2%. Third, yolo v2 proposed the use of predefined anchors instead of arbitrary anchors used by yolo v1. Motivated by the Faster R-CNN model, a set of 5 predefined anchors were proposed. The anchors were obtained by passing the training data through a k-means cluster. The best 5 clusters were used to define the size of the anchors. The IoU was used as a distance metric for the k-means cluster. By analyzing the obtained anchors, the anchors of real objects are not arbitrary and have a specific size and aspect ratio. Instead of predicting arbitrary bounding boxes, the offset of each anchor was predicted. If the offset values were constrained, the prediction diversity can be maintained and each prediction can focus on a specific shape. To generate bounding box prediction parameters (b_x, b_y, b_w, b_h, b_o), 5 parameters (t_x, t_y, t_w, t_h, t_o) were predicted then the sigma function was applied to constraint its possible offset range. The bounding box parameters can be computed as 5.

$$b_x = \sigma(t_x) + c_x \tag{5}$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$\sigma(t_o) = p_r(object) \times IoU\ (b, object)$$

(t_x, t_y, t_w, t_h, and t_o): Yolo generated predictions

(c_x, c_y): top left corner coordinate of the anchor

(p_w, p_h): width and the height of the anchor respectively

(b_x, b_y, b_w, and b_h): (x, y) coordinate, width, and heigh of the predicted bounding box respectively

$\sigma(t_o)$: box confidence score

Figure 3 present an illustration of the bounding ox predictions by predicting the offset value. This modification has made the initial training more stable and better localization performance was achieved.

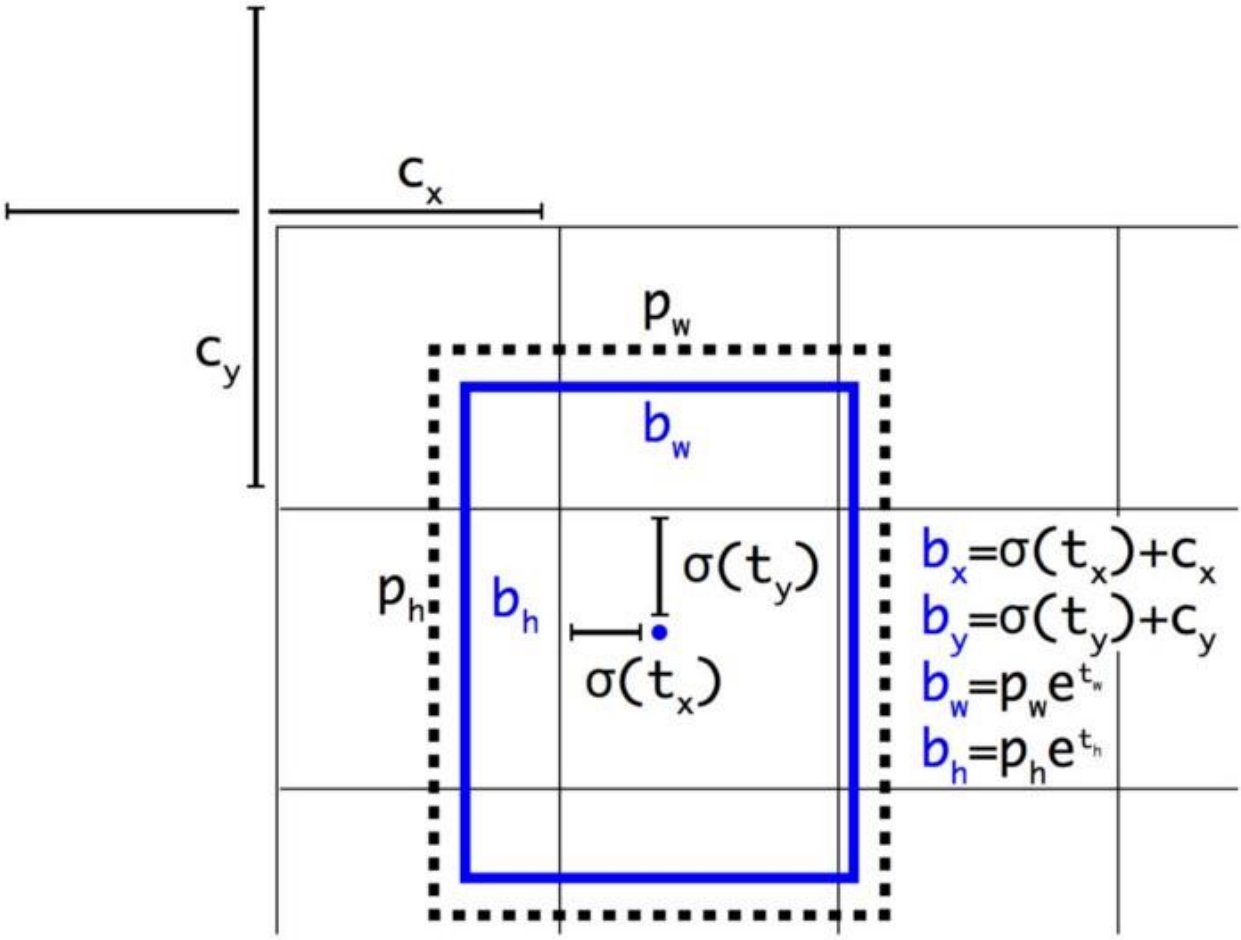

Figure 3: bounding box prediction in yolo v2 [14].

The class probability prediction has been moved from the cell level to the bounding box level. Figure 4 present the difference between yolo v1 and yolo v2 for the prediction generation.

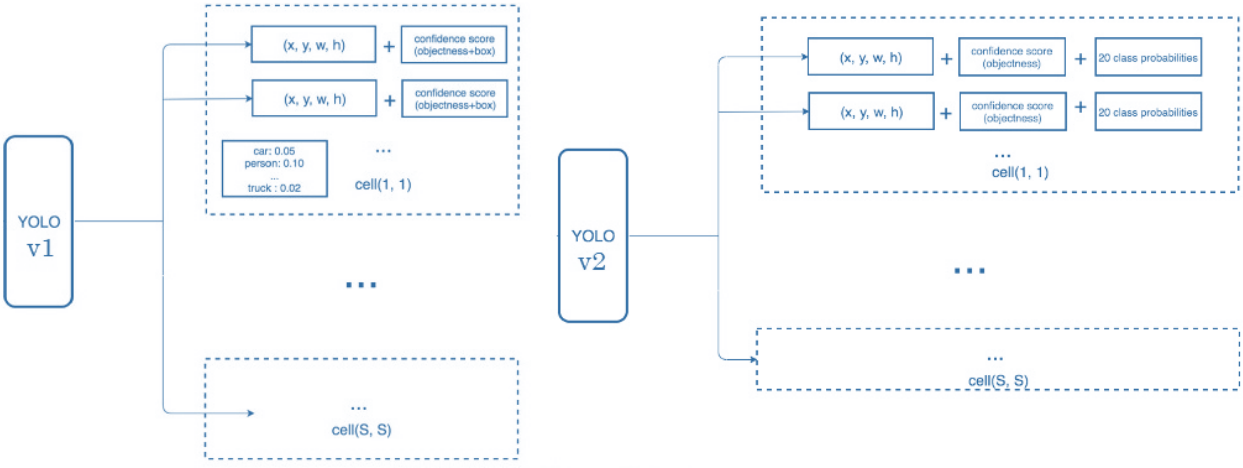

Figure 4: generating prediction in yolo v1 and yolo v2

To solve the problem of detecting small objects, yolo v2 proposes the use of fine-grained features. In effect, 26 × 26 × 512 layer was reshaped to 13 × 13 × 2048 then concatenated with the output layer 13 × 13 ×1024. After that, a convolution layer was applied on the new 13 × 13 ×3072 to generate final predictions. Finally, to gain more performance, a data augmentation technique was adopted by using images from different resolutions such as 320×320, 352×352, … and 608×608. For every 10 epochs, yolo v2 randomly selects a new image size. This technique allows detecting objects at different sizes and scales. The yolo v2 has achieved an mAP of 76.8% on the pascal voc 2007 and runs at 67 FPS on the Nvidia Titan X GPU. The yolo v2 has improved the accuracy but the model still struggling from detecting small objects.

3.3. Yolo v3

The yolo v3 was proposed to enhance the accuracy and allow detecting objects at different scales. To enhance the accuracy a better backbone was used with skip connection. The backbone was composed of 106 convolution layers and pooling layers were replaced with strided convolution layers. As proved in [31] using strided convolution instead of max-pooling layers can improve the accuracy without increasing the network complexity. 1x1 convolution kernels were used to compress the number of channels and 3x3 convolution kernels were used extract features. The proposed backbone has achieved the same accuracy as ResNet 152 [32] with fewer FLOPS and runs two times faster.

For class prediction, yolo v3 proposed to replace softmax layer independent logistic classifiers that predict the likeliness of the input belongs to a specific class. For the classification loss, yolo v3 proposed the use of the binary cross-entropy function instead of the old mean square error function used in the earlier versions. The proposed modifications have improved the accuracy and also reduces the computation complexity of the model.

To predict bounding boxes, yolo v3 used the same method proposed by yolo v2 with some modifications. A logistic recognition layer was used to compute the bounding box parameters. Yolo v3 changed the way of final prediction generation. If a certain anchor has a higher overlap with the ground truth bounding box than other anchors, then its corresponding objectness score is assigned to 1. Other anchors with an overlap greater than a predefined threshold are ignored and no more computation is performed. So, for each ground truth bounding box, only one anchor is assigned. To compute the loss, the (t_x, t_y) were used instead of (b_x, b_y).

For detecting objects at different scales, yolo v3 proposed features pyramid network [35] like. For each location of the feature map, three predictions are made each at a different scale. Three different scales were used, a scale for small objects, a scale for medium objects, and a scale for big objects. The detections were made at different feature map level. The first detection scale for big objects was performed on the last feature map. The second detection scale for medium objects was performed by going back two layers and upsampled by two. Then, a feature map with higher resolution and merge it with the upsampled feature map using element-wise addition. A convolution layer was applied to the result of the element-wise addition to generating the final prediction. The third detection scale for small objects was performed by going back another two layers and repeated the same process as the second scale. Figure 5 present the architecture of the yolo v3 model with the detection at different scales.

Yolo v3 has achieved state-of-the-art performance on the MS COCO dataset [19] with 33% of average precision. Compared to the single-shot multi-box detector SSD [22], it has similar precision but runs three times faster. The yolo v3 has significantly improved small objects detection compared to older versions and enhanced global precision while maintaining real-time processing conditions.

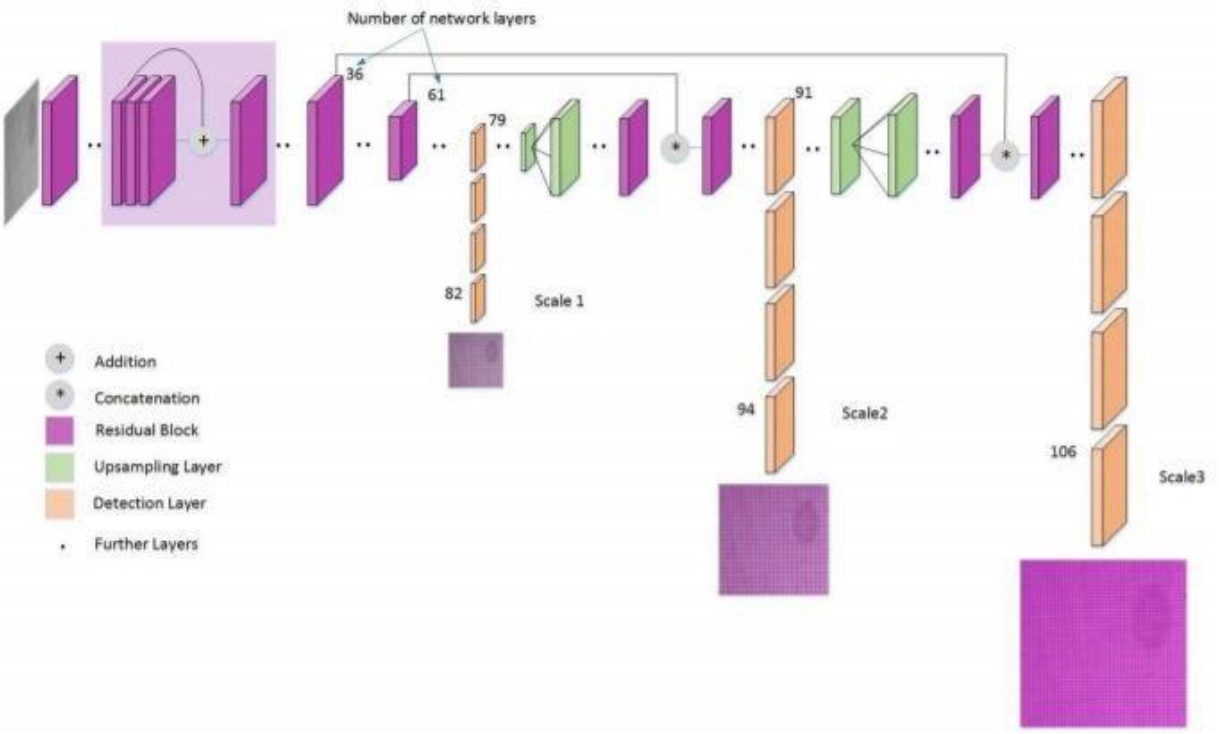

Figure 5: yolo v3 architecture.

3.4. Yolo v4

The yolo v4 proposed serious improvement over yolo v3. It has modified all parts of the detector to enhance both speed and accuracy while reducing computation complexity. First of all, yolo v4 has modified the backbone by changing the features extractor network and the optimization techniques. It started by proposing a set of techniques that enhance the training strategy without any cost to the inference. Second, the detector was modified by applying a set of techniques that allow enhancing the accuracy.

Data augmentation techniques were proposed to increase image variability in order to enhance the generalization power of the model in the training stage. For example, a photometric distortion technique was used by adjusting brightness, hue, contrast, saturation, and noise of the image to provide its varieties, the geometric distortion technique was also applied by rotating, flipping, random scaling, or cropping the image. Also, a mix-up technique was deployed to produce new images through mixing existing images based on weighted linear interpolation of two existing images. the cut mix technique was also adopted by concatenating random patches of different images where the ground truth has been mixed proportionally to the area of the patches. Mosaic data augmentation technique was applied by combining four training images in one image. The proposed data augmentation techniques were proposed to increase the robustness of the model against data corruption and to improve the generalization power.

In early CNN models, the dropout technique was applied in fully connected layers to avoid overfitting by forcing the model to learn from different features. However, this technique cannot be applied on convolution layers since neighbor neurons are highly correlated. In effect, even some neurons were eliminated, the spatial information still invariant. In yolo v4, a dropBlock technique was proposed to be applied to convolution layers. So, instead of dropping a single neuron a block of neurons with a fixed size is dropped. Figure 6 present the difference between dropout and dropBlock techniques.

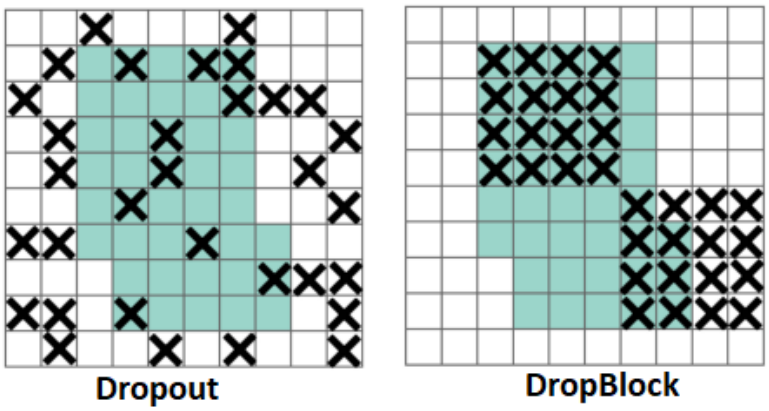

Figure 6: Difference between dropout and dropBlock techniques.

Non-linear activation functions have played a huge role in the success of deep learning by extending the learning capability of neural networks and allow the processing of more complex data. Many non-linear activation functions have been proposed such as the rectified linear units (ReLU) [37], its variants PReLU, FReLU [38] and leakyReLU [39] and swish [40]. The Mish function [41] has proved that it can achieve better performance compared to other non-linear activation functions. It is based on the hyperbolic tangent (tanh) activation function. The Mish function can be computed as 6.

$$f(x) = x \tanh(softplus(x)) \tag{6}$$

The Mish function is bounded below and unbounded above. A comparison between the most popular activation functions is presented in figure 7. It preserves a small negative amount (max value about -0.31) of the weight value to eliminate the drying problem of ReLU function. In effect, a large negative bias value saturates the ReLU function and stops weight updating in the backpropagation which makes the neuron useless for the prediction. In yolo v4 , the Mish function was used to enhance the information flow and to achieve a better regulation effect.

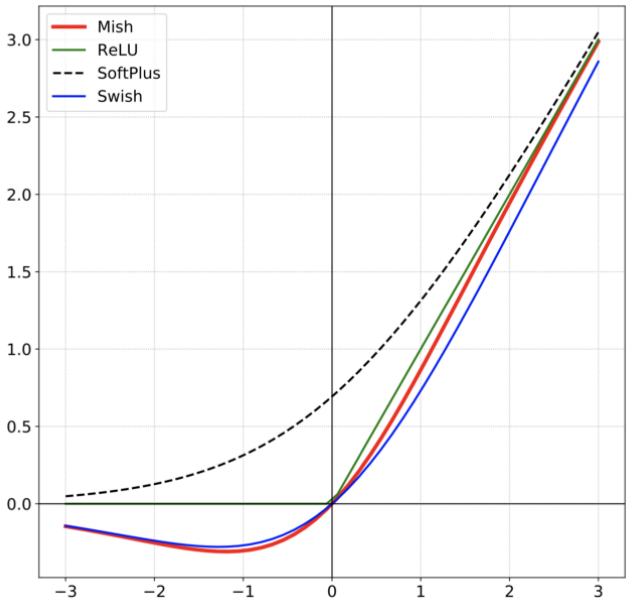

Figure 7: comparison between most popular non-linear activation functions.

A new backbone was proposed for yolo v4 to enhance the receptive field, increase the complexity and reduce the training difficulties. The Cross Stage Partial Darknet53 (CSPDarknet53) was used as a backbone, which is based on the combination of DenseNet model [33] and CSPNet model [34]. The main idea of the DenseNet model was the use of dense blocks which are composed of a set of layers composed each by a batch normalization layer a non-linear activation layer and a convolution layer. Each layer in the dense block takes as input all outputs of previous layers. so, each layer in the dense block generates four features maps. The DenseNet model was built by connecting multiple dense blocks using transition layers composed of convolution and pooling layers. Figure 7 present the architecture of the dense block and the DenseNet model. The CSPNet proposed to separate the input of the dense lock in two parts. The first part goes directly to the transition layer and the second part passed as normal by the dense block. The proposed modification has reduced the computation complexity.

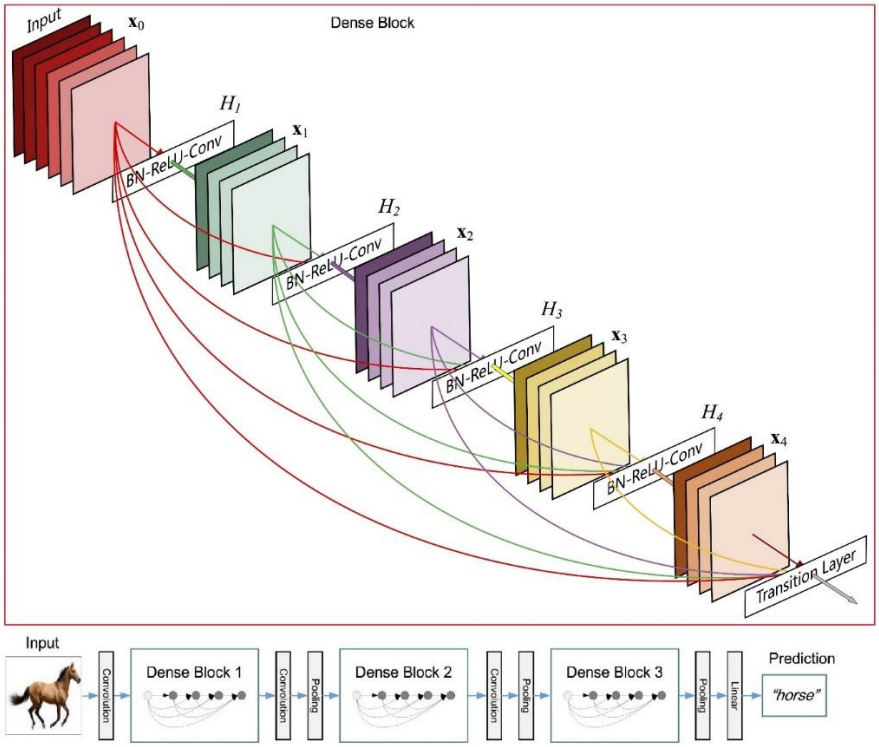

Figure 8: denseNet model and dense block architecture

In yolo v4, the concept of DenseNet and CSPNet were applied on the Darknet53 model resulting in the CSPDarknet53. The model has achieved better performance than ResNet [32]in object detection even it has lower classification performance.

As for yolo v3, yolo v4 investigated object detection at different scales. Thus, a hierarchy structure was proposed based on a modified features pyramid network (FPN) [35] to collect features at different spatial resolutions. The main concept of the FPN is the following: it starts by upsampling the previous top-down layer by 2, then an element-wise addition is performed with the bottom-up neighboring layer and finish up by passing the resultant features through a 3x3 convolution layer to reduce the upsampling artifact and create a new feature map. Figure 8 present the concept of the FPN. The FPN allows the model to extract features from different resolution to detect objects at different scales without any modification to the input image size or the structure if the feature extraction network.

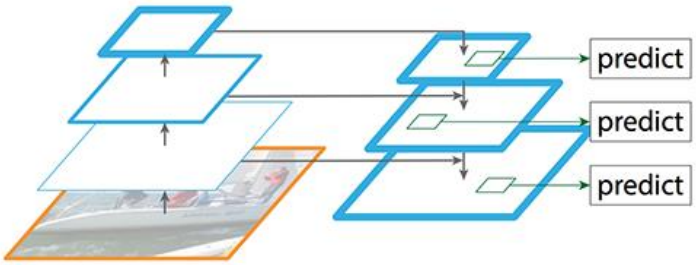

Figure 9: FPN architecture

Generally, fully connected layers are used to generate predictions but those layers have a fixed size and require an input with a fixed size. But for object detection tasks, detected objects have different sizes and cannot be passed to the same fully connected layers. this problem forces image scaling to a fixed size which results in degrading the accuracy in case of removing parts of the detected object. In addition, another problem commonly faced is the fixed size of the sliding window.

The SPPNet [36] has addressed the mentioned problems by using a spatial pyramid pooling layer. This layer was applied at the end of the features extraction network to generate the fixed size of features maps whatever the size of the generated feature maps to be connected to the fully connected layers. The spatial pyramid pooling layers is composed of four steps. First, each input feature map is pooled to become a one neuron layer. Second, each input feature map is pooled to became four neurons layer. Third, each input feature map is pooled to became 16 neurons layer. Finally, all the generated maps in the previous steps are concatenated to form a fixed size vector in order to connect it to the fully connected layers.

The spatial pyramid pooling layer was adopted by the yolo v4 to allow using sliding windows with different sizes on the same future map. The yolo v4 does not use fully connected layers and there is no need to fix the size of the feature maps. So, the feature maps generated by different kernel sizes were concatenated and used as input to the next layers. An illustration of the spatial pyramid pooling layer used by yolo v4 is presented in figure 9.

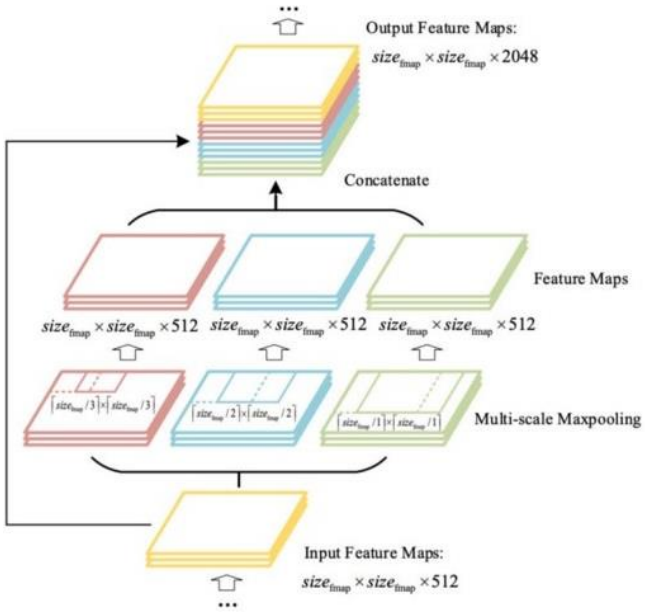

Figure 10: spatial pyramid pooling layer used in yolo v4

To gain more performance, the most recent models have introduced shortcut connections. Information flow through the network is considered the key decision of model design. The recent path aggregation network (PAN) for object detection has introduced a new shortcut connection that makes low layers features propagate to the top layers easier. PAN was based on FPN with a modified shortcut path. Figure 10 present the PAN architecture. In FPN, the spatial information passes through the network for more than 100 layers (red path in figure 10). In PAN, the special information passes through ten layers to reach the top of the network (green path figure 10). The proposed to add a new path that allows the network to take advantage of fine-grained information from low layers at top layers with fewer connections. To collect information at layers, PAN takes the all-previous layers features and add them to the current layer. Yolo v4 proposed to concatenate the features of all previous layers with the current layer instead of adding them.

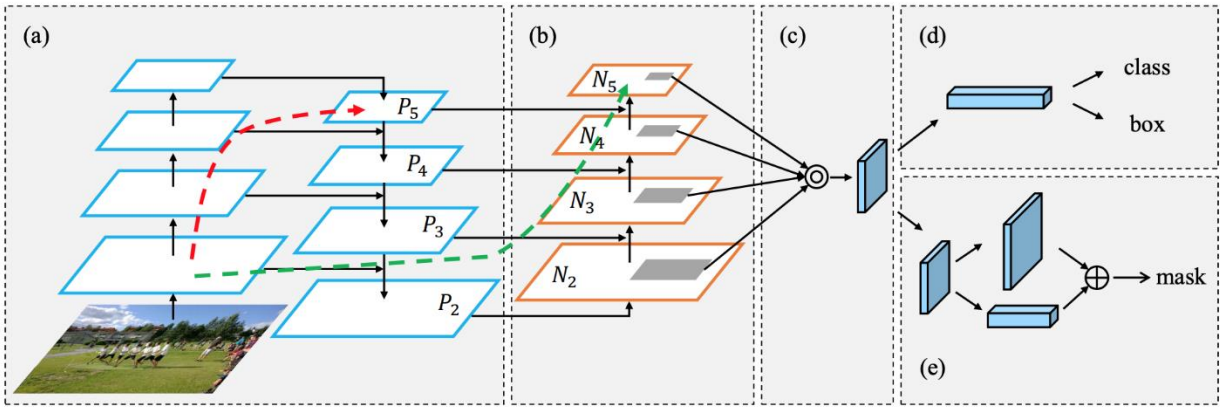

Figure 11: PAN architecture

Most recent CNN models have adopted spatial attention module to refine feature maps. A spatial attention module is composed of max-pooling and average pooling layers applied separately on feature maps to create two new feature maps. Then the new feature maps are feed into a convolution layer followed by a non-linear activation layer based on the sigmoid function to create the spatial attention map. In yolo v4 the spatial attention module was modified by eliminating the pooling layers and pass the feature maps through the convolution layer and the sigmoid layer only.

For the detector, yolo v4 has proposed a set of techniques to enhance the detection accuracy at training and inference. The main role of the detector is to generate predictions that contain the parameters of the predicted bounding box and the class of the detected object.

Batch normalization regularization [42] has been widely adopted in CNN models. It was used to collect the mean and variance of training samples for each mini-batch to bleach the layer input. But, in the case of using a small-size mini-batch, the mean and variance estimations become noisy. To address this problem, yolo v4 proposed the use of cross mini-batch normalization [43] that collects mean and variance estimations between mini-batches for every single batch. Also, it collects estimations of recent batches to enhance the estimations of the current batch.

To improve the training accuracy, yolo v4 proposed a self-adversarial training technique. The main idea of this technique is to make an adversarial attack by the model on itself. To do that, a simple forward on a training sample was performed. Normally, the model updates the weights to improve the accuracy in the backpropagation process. However, the model goes in the opposite direction by changing the training sample to degrade the accuracy as maximum as possible. Then the model is trained as normal using the

new training sample while using the old ground truth bounding box and class label. This technique allows to enhance the generalization power of the model and avoid the overfitting problem.

To eliminate the sensitivity of the predicted bounding box to the grid, yolo v4 proposed to modify the equation used to calculate the parameters of the bounding box by multiplying the sigma (σ) function by a factor greater than one. This modification was proposed in case that b_x value approaches from c_x or c_x+1 value. To achieve this condition t_x absolute value must be too high. To avoid this issue, the mentioned solution was proposed to eliminate the effect of the grid in the case of undetectable objects.

Multiple anchor boxes were used to detect the same object. Traditionally, CNN cannot predict a set of bounding boxes for the same object at different scales. So, in yolo v4 anchors were used to split the image space into different strategies. Based on the last convolution feature map created by the backbone, a set of anchors with different ratios were used to detect any object of any size. Then the IoU was used to decide which box fit better for the detected object according to a predefined threshold.

To provide the final prediction, non-maximum suppression (NMS) was used to delete the redundant bounding box and select the best-fit bounding box for the directed object. However, for occlusion cases, the NMS does not perform well. To achieve better results in challenging conditions, yolo v4 proposed to use the distance IoU (DIoU) [44] as an evaluation metric for the NMS technique. The DIoU-NMS measures the distance between the central point of each two predicted bounding boxes to decide which one is the best prediction.

To train a CNN model, a loss function that computes the difference between the target and predicted values must be optimized. The loss function provides information on how to update the weights to reduce the difference. Generally, the loss function gives the updating directions of the weights in case of wrong predictions but in the case of using the IoU as a loss within non-overlapped boxes with the ground truth, the loss function cannot decide which one is the best prediction and the weights cannot be updated correctly. To handle this problem, the complete-IoU (CIoU) loss [44] was used. The CIoU loss presents two advantages compared to the IoU loss. First, the use of the central point distance concept that computes the distance between the bounding boxes based on its central points. Second, it compares the aspect ratio of the ground truth bounding box and the predicted bounding box to decide if the prediction is considered or ignored. Based on those concepts, the CIoU loss defines the quality of the prediction. The CIoU can be computed as 7.

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b,b^{gt})}{c^2} + \alpha v \tag{7}$$

$$v = \frac{4}{\pi^2}(\arctan\left(\frac{w^{gt}}{h^{gt}}\right) - \arctan(\frac{w}{h})^2$$

Where b is the central point of the predicted bounding box, $b^{gt}$is the central point of the ground truth bounding box, c is the diagonal length of the smallest overlap box between two bounding boxes, α is a positive trade-off parameter, $\rho(.)$ is the Euclidian distance and v is a function that measures the difference between aspect ratios between two bounding boxes.

The focal loss proposed by the RetinaNet model [47] was deployed to avoid the class imbalance problem. The focal loss was proposed to address the scenario of having an extreme difference between background and foreground samples during the training process. Usually, the cross-entropy loss function is used to train classification models by penalizing the error strongly if the class probability is high. However, for classes with a limited amount of training samples, the cross-entropy can penalize true positive samples and degrade

the classification accuracy of those classes. The focal loss function was proposed to overcome this problem by introducing a coefficient to force the model to focus on low accuracy classes. The focal loss function can be computed as 8.

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)) \tag{8}$$

$$p_t = \begin{cases} p, & y = 1 \\ 1-p, otherwise \end{cases}$$

$$\alpha_t = \begin{cases} \alpha, & y = 1 \\ 1-\alpha, otherwise \end{cases}$$

The α_t was used to control the weight of positive and negative samples and the $(1-p_t)^{\gamma}$ was used to consider hard and easy samples. α is a hyperparameter used for cross-validation.

The learning rate decides on the conversion speed and quality. The traditional backpropagation technique uses a fixed learning rate during the training process. In yolo v4, the cosine annealing scheduler [45] was proposed to adjust the learning rate according to the convergence process. This technique starts with a slow reducing rate then a quick reducing rate is applied until reaching a very small learning rate at the end of the training. The use of this technique makes the training more stable and a better minimum can be achieved by avoiding local minimums.

Finding the best hyperparameters for a CNN model is a very challenging task. Initializing the model with arbitrary hyperparameters can lead to low accuracy or training difficulties. To solve this problem, yolo v4 proposes to use the genetic algorithm to find the best hyperparameters for its model. The genetic algorithm generates arbitrary hyperparameters and uses them to train a set of models. The top best performance is selected and a set of slightly mutated hyperparameters are generated according to the original ones. The same step is repeated until finding the ultimate hyperparameters are discovered.

In summary, the yolo v4 has proposed many improvements for backbone, detector, and training strategy. The architecture of the yolo v4 is illustrated in figure 12. Yolo v4 has achieved state-of-art object detection on the MS COCO dataset with an average precision of 43.5% and an inference speed of 65 FPS on the Nvidia Tesla V100 GPU. It has achieved an improvement of 10% in terms of precision and 12% in terms of inference speed compared to the yolo v3. Besides, the model has become trainable on a single GPU easier and faster.

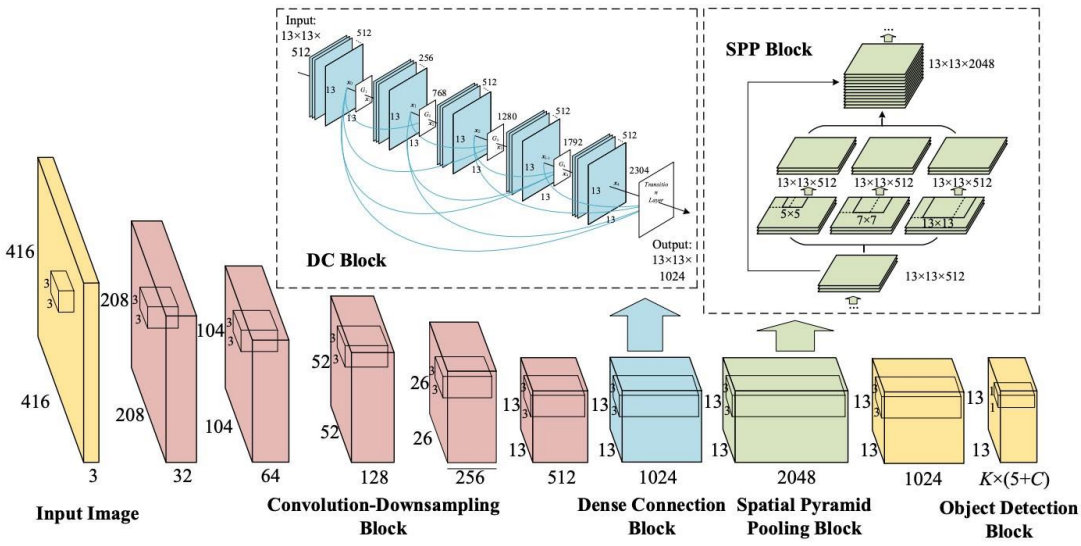

Figure 12: yolo v4 architecture

3.5. Yolo v5

The yolo v5 was proposed to make the model suitable for edge devices. The model has the same architecture as yolo v4 with minor modifications. First, the backbone was modified by replacing the CSPDarknet53 with CSPNet [53]. The CSPNet is lighter than the CSPDarknet53 with similar performance. Second, the activation function was modified and the sigmoid weighted linear units (SiLU) function was used. The SiLU function can be computed as 9.

$$f(x) = x\sigma(x) \tag{9}$$

The new activation function has extended the learning capability of the neurons and improved the precision. The curve of the SiLU function compared to the ReLU function is presented in figure 13.

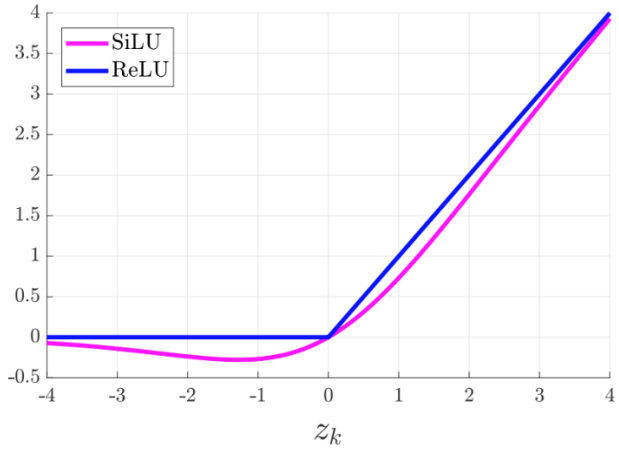

Figure 13: comparison between SiLu function and ReLU function

To compute the loss, yolo v5 proposed the use of the Binary Cross-Entropy with Logits Loss function. And maintains the same loss function for the localization proposed in yolo v4. The focal loss has been deployed

ds in yolo v4. For yolo v5, 16-bits floating-point representation was used for weights and activation in both training and inference. This modification has significantly accelerated the processing speed.

The yolo v5 model has achieved an average precision of 50% on the MS COCO dataset with an inference speed of 140 FPS on the Nvidia Tesla V100 GPU. another improvement over the yolo v4, that the yolo v5 model size is light with a size of 27 MB for its lite version yolo v5s compared to 244 MB for the yolo v4 model. The small size of the model makes it suitable for implementation on mobile devices such as smartphones and embedded devices.

4. Experiments and results

In this work all models were tested using a desktop with a Linux operating system equipped with an Intel i7 CPU, 32 GB of RAM, and an Nvidia GTX 960 GPU. all models were developed based on the Pytorch deep learning framework with GPU acceleration through the CUDA and cuDNN libraries. The open cv library was used to load and display images.

To train the models for outdoor object detection, the BDD100K dataset [48] was used. The dataset was collected in the united states using a smartphone. Images were captured at real-world conditions such as different weather conditions, different lighting conditions, and hard situations like occlusion. As it was named, the dataset consists of 100K videos. Each video has 40 seconds long with a resolution of 720p and a frequency of 30 FPS. The GPS/IMU information was added for each video. The dataset was designed for multiple tasks such as image tagging, road object detection, lane marking, and instance segmentation. In this work, the dataset was used for road object detection. For this task, the images were labeled with bounding boxes where each box was assigned with a class label. The instance distribution of each class is presented in figure 14. The number of instances per class was not balanced. For example, there are 16505 instances for bus and 179 instances for train. The dataset was divided into a training set and testing set where 70% of data was used as a training set and the remain 30% was used as a test set. For both training and testing, the images were resized to 608x608.

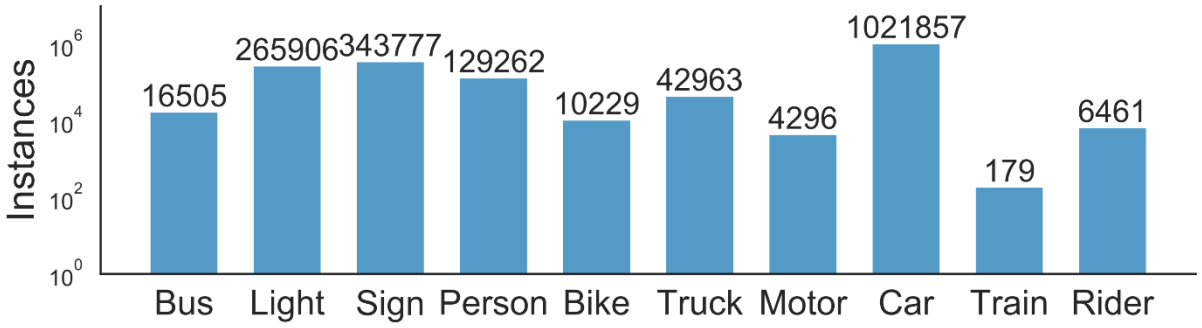

Figure 14: Instance distribution of the BDD100K dataset for road object detection

All models' weights were initialized using pre-trained weights on the MS COCO dataset. the transfer learning technique was applied by fine-tuning the models with the BDD100K dataset. For the evaluation of accuracy performance, the mean average precision (mAP) metric proposed by MSCOCO was used. The models were evaluated in terms of precision, the number of parameters, number of FLOPS, model size, and inference speed.

The models were trained using different optimizers. For yolo v1, yolo v2, and yolo v3, the gradient descent with momentum [49] was used with a learning rate of 0.001 and weight decay of 0.005. yolo v4 and yolo

v5 were trained using the Adaptive Moment Estimation (Adam) optimizer [50]. The Adam optimizer does not need an initial learning rate since it generates a random learning rate and optimizes it along with model training.

After training and testing, the obtained average precision (AP) of each model per class is presented in table 1. The reported results show an overview on how each model performs for detecting each class. As presented in table 1, yolo v1 and v2 struggle in detecting small objects like traffic signs and lights. Besides, those models fail in detecting close objects such as pedestrian s in crowded areas. The AP of pedestrians is lower than riders which are both human beings but the main difference is in the shape and scale. The existence of crowded areas has limited the performance of the models in detecting pedestrians.

Table 1: Obtained average precisions of each model per class

Model	AP (%)									
	Bus	Truck	Train	Car	Motor	Bike	Rider	Person	Light	sign
Yolov1	80.35	82.93	62.35	74.36	65.45	62.96	68.35	63.77	38.47	47.89
Yolov2	84.56	88.45	70.49	79.57	68.95	66.19	74.52	67.94	44.94	51.03
Yolov3	87.64	89.94	73.83	84.66	70.02	69.37	77.85	70.22	57.89	60.34
Yolov4	89.47	90.56	80.47	86.38	72.85	73.84	79.37	74.95	70.34	73.71
Yolov5	89.98	91.04	83.56	87.95	77.58	75.67	80.25	79.45	72.47	74.17

Besides, for yolo v1 and v2, objects with lower training instances have lower precision. For example, the train class has a lower precision with a big margin compared to truck and bus. In yolo v3, the problem of small object detection was fixed and higher precision was achieved. The problem of class imbalance was fixed by the yolo v4 through the use of the focal loss. The detection precision of the train class has been boosted compared to the precision achieved by yolo v3. The overall detection precision has been enhanced by the yolo v5 compared to the previous version.

Precision is not the only factor to decide on which model performs better. A summary of the evaluation results of each model were presented in Table 2. The reported results include the mAP, the number parameters, the number of FLOPS, the model size and the inference speed.

Table 2: summary of evaluation results of the yolo versions

Model	mAP (%)	Parameters (million)	FLOPS (billion)	Size (MB)	Speed (FPS)
Yolo v1	64.68	402.8	40.19	663	30
Yolo v2	69.66	439.7	62.94	254	25
Yolo v3	74.17	457.6	65.86	232	16
Yolo v4	79.19	508.3	72.46	246	21
Yolo v5x	81.2	_87.7	10.05	168	27

As shown in Table 2, the yolo v5 has the best trade-off between inference speed and precision in addition to having the smallest model size. The yolo v1 has low precision and a very large model size but it was the fastest in terms of processing speed. Yolo v2 does not present much difference compared to yolo v1. Yolo v3 has enhanced the precision but decreased the inference speed. Compared to the previous version, yolo v4 has improved both precision and speed but its model size was increased.

Figure 15 present a chart for comparisons of the different evaluation factors. In real-world applications, real-time processing is a critical factor that decides on the efficiency of the model. However, for applications like ADAS, the computation complexity, and memory consumption are considered very important factors that judge the model efficiency for the desired application. As shown in figure 15, most of the yolo models except the yolo v5 has a large memory occupation size and need a lot of computation effort to achieve real-time processing. The yolo v4 has good precision and processing speed but requires 246 MB of memory. Such amount of memory is not available on embedded devices designed for ADAS. Even in the case of using a high-performance embedded device such as the Nvidia Drive AGX Pegasus [51], the needed amount is too high. The yolo v5 has achieved the best precision and a good inference speed with a model size lower than all the previous versions. The obtained results of yolo v5 make it useful for many applications including ADAS.

Based on the obtained results, yolo v5 is considered the best solution for outdoor object detection in urban spaces. The model was pretended to be integrated for ADAS which is equipped with limited computation resources. The model that has the lowest size with good accuracy is the best fit.

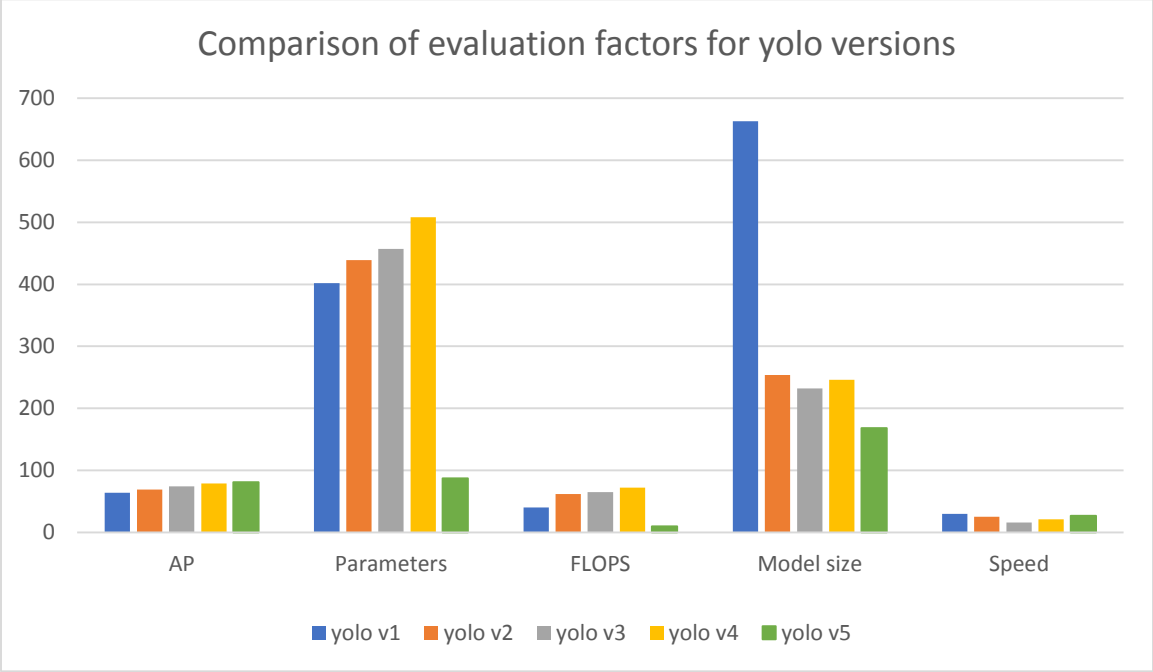

Figure 15: comparison between different evaluation metrics of the yolo models

5. Conclusions

Outdoor object detection is an important application for ADAS to elevate road safety. In this paper, we proposed to evaluate the performance of yolo different versions to decide which one fits better for outdoor object detection. the yolo models were designed to achieve real-time processing while getting good detection precision. The background of each version was detailed and the difference between them was highlighted. The BDD100K dataset was proposed for the training and evaluation of the models for outdoor object detection. Many experiments were performed and different results were reported. Different evaluation factors were considered such as the mAP, the number parameters, the number of FLOPS, the model size, and the inference speed to figure out which model performs well for the studied task. After, analyzing different evaluation factors, we decided that the yolo v5 is the best model for outdoor object detection applied for ADAS due to its high detection precision, fast processing speed, and low model size.

The yolo v5 has proved its efficiency in detecting different objects compared to other versions that suffer from many problems such as low precision at detecting small objects for yolo v1 and V2, class imbalance for yolo v3, and the high memory occupation of the yolo v4.

Acknowledgment: Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Small Groups. (Project under grant number 367/43).

Availability of data and materials:

All data analyzed during this study are properly cited by reference [48].

Competing interests:

The authors declare that there is no conflict of interest or competing interest regarding the publication of this article.

Funding:

Deanship of Scientific Research at King Khalid University.

Authors' contributions:

Riadh Ayachi is responsible for data collection and experimentation, Mouna Afif prepared the draft of the manuscript, Yahia said interrupt the results, and Abdessalem Ben Abdelali is the project manager.

Acknowledgements:

Not applicable.

References

[1] Hongtao, Lu, and Zhang Qinchuan. "Applications of deep convolutional neural network in computer vision." Journal of Data Acquisition and Processing 31, no. 1 (2016): 1-17.

[2] Afif, Mouna, Riadh Ayachi, Yahia Said, Edwige Pissaloux, and Mohamed Atri. "Indoor image recognition and classification via deep convolutional neural network." In International conference on the Sciences of Electronics, Technologies of Information and Telecommunications, pp. 364-371. Springer, Cham, 2018.

[3] Afif, Mouna, Riadh Ayachi, Yahia Said, Edwige Pissaloux, and Mohamed Atri. "Indoor object c1assification for autonomous navigation assistance based on deep CNN model." In 2019 IEEE International Symposium on Measurements & Networking (M&N), pp. 1-4. IEEE, 2019.

[4] Dhillon, Anamika, and Gyanendra K. Verma. "Convolutional neural network: a review of models, methodologies and applications to object detection." Progress in Artificial Intelligence 9, no. 2 (2020): 85-112.

[5] Zhang, Gang, Tao Lei, Yi Cui, and Ping Jiang. "A dual-path and lightweight convolutional neural network for high-resolution aerial image segmentation." ISPRS International Journal of Geo-Information 8, no. 12 (2019): 582.

[6] Afif, Mouna, Riadh Ayachi, Yahia Said, and Mohamed Atri. "Deep Learning Based Application for Indoor Scene Recognition." Neural Processing Letters (2020): 1-11.

[7] Afif, Mouna, Riadh Ayachi, Edwige Pissaloux, Yahia Said, and Mohamed Atri. "Indoor objects detection and recognition for an ICT mobility assistance of visually impaired people." Multimedia Tools and Applications (2020): 1-18.

[8] Gupta, Any, and Ayesha Choudhary. "A Framework for Traffic Light Detection and Recognition using Deep Learning and Grassmann Manifolds." In 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 600-605. IEEE, 2019.
[9] Ayachi, Riadh., Yahia Elfahem Said, and Mohamed Atri. "To perform road signs recognition for autonomous vehicles using cascaded deep learning pipeline." Artificial Intelligence Advances 1, no. 1 (2019): 1-58.
[10] Ayachi, Riadh, Mouna Afif, Yahia Said, and Mohamed Atri. "Traffic signs detection for real-world application of an advanced drivier assistancesystem using deep learning." Neural Processing Letters 51, no. 1 (2020): 837-851.
[11] Ayachi, Riadh, Yahia Said, and Abdessalem Ben Abdelaali. "Pedestrian Detection Based on Light-Weighted Separable Convolution for Advanced Driver Assistance Systems." Neural Processing Letters (2020): 1-14.
[12] Ayachi, Riadh, Mouna Afif, Yahia Said, and Abdessalem Ben Abdelaali. "pedestrian detection for advanced drivier assistancesystem: a transfer learning approach." In 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), pp. 1-5. IEEE, 2020.
[13] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.
[14] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263-7271. 2017.
[15] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
[16] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv preprint arXiv:2004.10934 (2020).
[17] YOLOv5 New Version - Improvements And Evaluation. Available at: https://blog.roboflow.com/yolov5-improvements-and-evaluation/
[18] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." International journal of computer vision 88, no. 2 (2010): 303-338.
[19] Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In European conference on computer vision, pp. 740-755. Springer, Cham, 2014.
[20] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.
[21] Ning, Chengcheng, Huajun Zhou, Yan Song, and Jinhui Tang. "Inception single shot multibox detector for object detection." In 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 549-554. IEEE, 2017.
[22] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.
[23] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

[24] Wang, Xiaoyan, Hai-Miao Hu, and Yugui Zhang. "Pedestrian Detection Based on Spatial Attention Module for Outdoor Video Surveillance." In 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), pp. 247-251. IEEE, 2019.
[25] Zhou, Zhigong, Xiaosong Lan, Shuxiao Li, Chengfei Zhu, and Hongxing Chang. "Feature Pyramid SSD: Outdoor Object Detection Algorithm for Blind People." In 2019 IEEE 5th International Conference on Computer and Communications (ICCC), pp. 650-654. IEEE, 2019.
[26] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.
[27] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117-2125. 2017.
[28] Zhang, Zhimeng, Jianan Wu, Xuan Zhang, and Chi Zhang. "Multi-target, multi-camera tracking by hierarchical clustering: recent progress on DukeMTMC Project." arXiv preprint arXiv:1712.09531 (2017).
[29] Morera, Ángel, Ángel Sánchez, A. Belén Moreno, Ángel D. Sappa, and José F. Vélez. "SSD vs. YOLO for detection of outdoor urban advertising panels under multiple variabilities." Sensors 20, no. 16 (2020): 4587.
[30] Algredo-Badillo, Ignacio, Luis Alberto Morales-Rosales, Carlos Arturo Hernandez-Gracidas, Juan Crescenciano Cruz-Victoria, Daniel Pacheco-Bautista, and Miguel Morales-Sandoval. "Real time FPGA-ANN architecture for outdoor obstacle detection focused in road safety." Journal of Intelligent & Fuzzy Systems 36, no. 5 (2019): 4425-4436.
[31] Ayachi, Riadh, Mouna Afif, Yahia Said, and Mohamed Atri. "Strided convolution instead of max pooling for memory efficiency of convolutional neural networks." In International conference on the Sciences of Electronics, Technologies of Information and Telecommunications, pp. 234-243. Springer, Cham, 2018.
[32] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.
[33] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708. 2017.
[34] Wang, Chien-Yao, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. "CSPNet: A new backbone that can enhance learning capability of CNN." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp. 390-391. 2020.
[35] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117-2125. 2017.
[36] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37, no. 9 (2015): 1904-1916.
[37] Eckle, Konstantin, and Johannes Schmidt-Hieber. "A comparison of deep networks with ReLU activation function and linear spline-type methods." Neural Networks 110 (2019): 232-242.
[38] Qiu, Suo, Xiangmin Xu, and Bolun Cai. "Frelu: Flexible rectified linear units for improving convolutional neural networks." In 2018 24th International Conference on Pattern Recognition (ICPR), pp. 1223-1228. IEEE, 2018.

[39] Zhang, Xiaohu, Yuexian Zou, and Wei Shi. "Dilated convolution neural network with LeakyReLU for environmental sound classification." In 2017 22nd International Conference on Digital Signal Processing (DSP), pp. 1-5. IEEE, 2017.
[40] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).
[41] Misra, Diganta. "Mish: A self regularized non-monotonic neural activation function." arXiv preprint arXiv:1908.08681 4 (2019).
[42] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In International conference on machine learning, pp. 448-456. PMLR, 2015.
[43] Yao, Zhuliang, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. "Cross-iteration batch normalization." arXiv preprint arXiv:2002.05712 (2020).
[44] Zheng, Zhaohui, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. "Distance-IoU loss: Faster and better learning for bounding box regression." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 12993-13000. 2020.
[45] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." arXiv preprint arXiv:1608.03983 (2016).
[46] Elfwing, Stefan, Eiji Uchibe, and Kenji Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning." Neural Networks 107 (2018): 3-11.
[47] Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In Proceedings of the IEEE international conference on computer vision, pp. 2980-2988. 2017.
[48] Yu, Fisher, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2636-2645. 2020.
[49] Yu, Hao, Rong Jin, and Sen Yang. "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization." In International Conference on Machine Learning, pp. 7184-7193. PMLR, 2019.
[50] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
[51] NVIDIA DRIVE™ AGX embedded supercomputing platforms. Available at: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/ Last accessed: 01/02/2021