



CONE DETECTION WITH CONVOLUTIONAL NEURAL NETWORKS FOR AN AUTONOMOUS FORMULA STUDENT RACE CAR

Laíza Milena Scheid Parizotto

Eric Aislan Antonelo

Federal University of Santa Catarina - UFSC, Florianópolis - SC, Brazil
laizamsparizotto@gmail.com, eric.antonelo@ufsc.br

Abstract. *Specific cones are used to delimit the tracks for the race cars in the Formula Student Driverless competition. To effectively race autonomously, the car must accurately detect them. In this context, this work investigates state-of-the-art Convolutional Neural Networks, specifically the so called You Only Look Once (YOLO) net, for robust and fast detection of cones from images of a camera mounted on the race car. To train YOLO, the Formula Student Objects in Context (FSOCO) dataset with four different classes of cones in the track is employed. The mean Average Precision (mAP) and network inference time are used to evaluate: (1) the influence of the image resolution; (2) the impact under different image perturbations such as brightness, exposure, blur, and noise; (3) the benefit of extra data augmentation for improving robustness to the aforementioned perturbations and out-of-distribution disturbances. We have found that: YOLO is a strong candidate for real-time cone detection for race cars; mAP increases with the image resolution, with just a slight increase in the network inference time; extra data augmentation for network training is beneficial for recovering the lost mAP when perturbations (of brightness, blur and noise) and especially out-of-distribution disturbances are applied to the images.*

Keywords: *autonomous race car, convolutional neural networks, cone detection, YOLO, Formula Student Driverless*

1. INTRODUCTION

One of the reasons for promoting autonomous race car competitions is that it provides an opportunity for developing and testing high-performance autonomous vehicles while advancing the state-of-the-art in the field. Formula Student Driverless is one of such competitions, held by Formula SAE¹ and that requires the development of a fully autonomous race car, where both speed and precision are important for successfully maneuvering it throughout the track, avoiding collisions which incur penalties and with the final goal of winning the competition.

To develop an autonomous vehicle, it is necessary to integrate different modalities of perceptions, such as cameras and other sensors, to a motion estimation and mapping pipeline and finally to an effective control system (Dhall *et al.*, 2019). For the competition, the autonomous race car needs to be able to correctly perceive its surroundings, for instance, assigning meanings to each of the identified objects in an image acquired from its camera. As the environment is a race track delimited by different traffic cones of distinct size and color, the race car needs to accurately detect cones from camera images, as well as their associated color, such as blue, yellow and orange with as little latency as possible and with minimum utilization of computational resources (Kabzan *et al.*, 2019). After that, one needs to estimate the 3D position of cones from images to infer their distances from the vehicle (Dhall *et al.*, 2019).

Object detection through Convolutional Neural Networks (CNNs) is the most promising approach since the AlexNet's CNN won the ImageNet visual recognition challenge in 2012 (Russakovsky *et al.*, 2015). This is because of its inherent visual processing capabilities given by its adaptive convolution layers, which are basically filters that can be trained through labeled data. In this sense, CNNs are trained to extract the most relevant features for a certain visual task, conditioned on the available labeled images. CNN architectures such as YOLO (Bochkovskiy *et al.*, 2020) and Faster Region Based Convolutional Neural Networks (R-CNN) (Ren *et al.*, 2015) are the most promising approaches for real-time object detection. Considering that speed is a crucial requirement for race cars, in this work, we focus on YOLO-based object detection because of its superiority in terms of inference time.

Successfully deploying supervised learning models, such as CNNs, depends on many factors, one of which is the amount of available high-quality labeled data. In our context of race car competitions, two datasets are mostly used for cone detection: MIT dataset (Strobel *et al.*, 2020) and FSOCO dataset, (Dodel *et al.*, 2020). The latter is the most recent dataset and also more relevant to the race car environment as it contains cone images taken from realistic race cars on

¹Society of Automotive Engineers

various real race tracks, unlike the former which contains any type of traffic cone (e.g., from an urban street) and does not specify the classes of cones that are specific to the competition.

Although most of the previous work on autonomous driving presents results that integrate perception to motion estimation and control (Caporale *et al.*, 2019; Kabzan *et al.*, 2019; Tian *et al.*, 2018; Chen *et al.*, 2019), as far as the authors know, none of them have done a detailed study on cone detection using two of the latest versions of YOLO and the newly introduced FSOCO dataset. In this work, we aim to show a more in-depth investigation of different versions of YOLO for cone detection, taking into account detection accuracy and inference time. We show that a simpler architecture of YOLO with higher image resolution is faster while still maintaining most of the accuracy of the full architecture. Besides, our results show that investing in data augmentation is much needed for detection robustness, especially in out-of-distribution image disturbances.

1.1 Related Work

Perception for Formula Student Driverless race cars. In de la Iglesia Valls *et al.* (2018), the first autonomous race car to win the competition is presented. They describe not only the perception system, but also the state estimation method and other system integration issues. Cone detection is achieved using mainly 3D Light Detection And Ranging (LiDAR) sensors, while no details have been given concerning a camera-based solution. Dhall *et al.* (2019) presents a cone detection system for monocular cameras using their own manually annotated dataset of cone images. They employ version 2 of YOLO(v2) for detecting cones from single images, and also present methods for recovering their 3D position, but do not investigate in-depth cone detection. (Strobel *et al.*, 2020) also describes a complete visual perception system for a race car with podium finishes at all competitions for which it raced. For cone detection, they employ YOLOv3 and also provide a publicly available dataset of traffic cones. The newest and most high-quality dataset for cone detection for the competition, FSOCO, is presented in (Dodel *et al.*, 2020), which also shows some preliminary results on cone detection using YOLOv4.

Evaluating robustness to environmental changes in autonomous driving. Some studies investigate robustness of CNNs to common corruptions. Shafiee *et al.* (2021) indicates that generating different test cases that leverages real-world changes in driving conditions like rain, fog, snow, and lighting conditions are important in the evaluation of autonomous driving systems. Michaelis *et al.* (2019) considers three different datasets with a large variety of image corruptions and shows that network performance decreases by 30% to 60%, and it also applies an augmentation technique to increase robustness. Dodge and Karam (2016) evaluates five types of quality distortions: blur, noise, contrast, JPEG, and JPEG2000 compression, and shows that state-of-art networks are susceptible to these quality distortions, especially to blur and noise.

1.2 Contribution

This work investigates cone detection approaches for the perception module of race cars, taking into account inference time, accuracy and robustness to perturbations, and has the following contributions: (1) it provides results on different versions of the YOLO architecture for detection of coloured traffic cones belonging to four possible classes, using the newly introduced FSOCO dataset; (2) It shows the influence of the image resolution for achieving a trade-off between accuracy and inference speed; (3) It presents the impact under different image perturbations such as brightness, exposure, blur, and noise; (4) It reveals the benefit of extra data augmentation for improving robustness to the aforementioned perturbations and out-of-distribution disturbances.

2. METHODS

2.1 Convolutional Neural Networks (CNN)

CNNs are neural networks that use convolution in place of general matrix multiplication in at least one of their layers (Goodfellow *et al.*, 2016). A convolution operation on a digital image (Fig. 1 - right) basically consists of sliding a kernel (weight matrix) throughout the image (input) while performing a dot-product operation between the kernel and the corresponding pixels in the image in order to generate convolved features in an output layer (Aggarwal, 2018; Goodfellow *et al.*, 2016). For each layer in the CNN, this procedure is done considering different kernels, which results in multiple output activation tensors called feature maps. They represent the result of different filters applied on the input image that learn to detect specific visual patterns after training the whole network. The number of feature maps (or applied filters) is represented by the depth of a layer which usually grows as one deepens through the network layers (Fig. 1 - left). The deepest convolutional layers can represent more abstract concepts from the image.

After the convolution, an activation function, usually the Rectified Linear Unit (ReLU), is applied to generate the final value of feature maps. In addition, CNNs can have pooling layers that are interleaved with the convolution layers (Aggarwal, 2018). The pooling operation produces another layer with the same depth (i.e., same number of feature maps in the respective layer) and it is used to reduce the spatial dimensions of each feature map. Unlike convolution, each

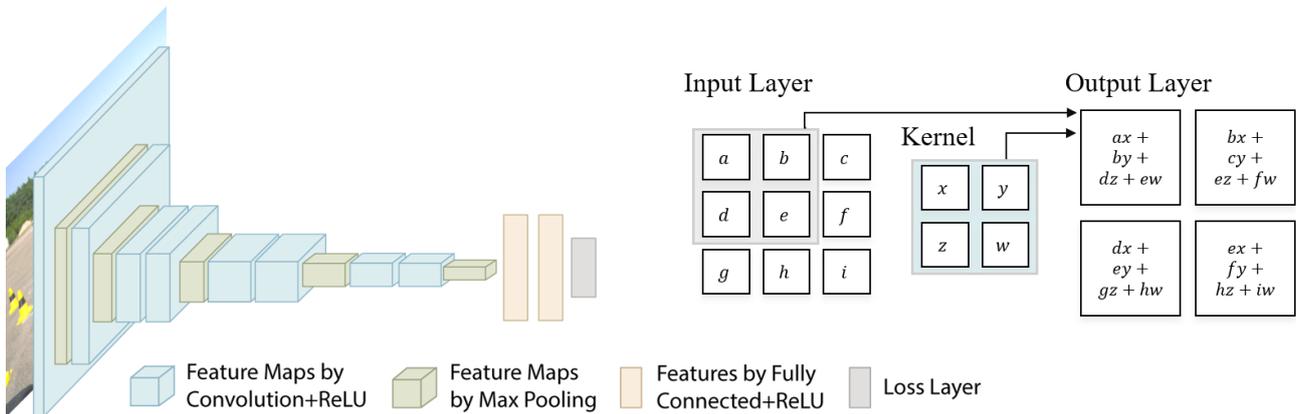


Figure 1: Left: A typical architecture of a CNN with convolution layers, pooling layers and fully connected layers: VGGNet from Simonyan and Zisserman (2015). Right: A 2D convolution between a 3×3 input and a 2×2 kernel, resulting in a 2×2 feature map.

feature map is independently processed and therefore the number of output feature maps is exactly equal to the number of input feature maps when pooling is applied (Aggarwal, 2018). Usually, max pooling is the chosen operation, which takes the maximum value of a square region in the image as output. Pooling yields robustness to small variations in the pixels of the input image. The last convolutional/pooling layer in a CNN connects to one or more fully connected layers, which receive the high-level features extracted from visual images, projecting them into an output space that is dependent on the desired task, e.g., regression or classification. The training procedure of the whole network is done through stochastic gradient descent in an iterative way (Goodfellow *et al.*, 2016).

2.2 Object Detection

A fundamental challenge in computer vision is the ability to classify objects in images and determine their spatial location. There are different approaches to object recognition (Liu *et al.*, 2019), such as: (1) classification or categorization aims to identify what type of objects are presented in an image, without necessarily localizing them; (2) object detection consists of finding (predicting) the bounding boxes of objects in a image together with their respective classes; and (3) semantic segmentation corresponds to assigning a class to every pixel in an image. Other types of segmentation include: instance segmentation (Liu *et al.*, 2019), panoptic segmentation and dense pose prediction (Wu *et al.*, 2019).

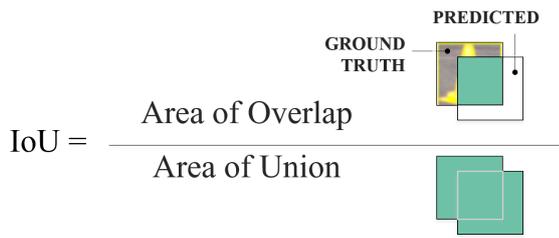
Finding models that are highly accurate and also fast in their inference is not trivial (Liu *et al.*, 2019). Usually, one needs to make a trade-off between these two metrics according to the application. For instance, in the *two-stage approach*, implemented by Faster R-CNN (Ren *et al.*, 2015), which is highly accurate, is based on a sequential procedure of first removing the background and only then classifying the remaining regions (Liu *et al.*, 2019). On the other hand, the *one-stage approach*, e.g. the Single Shot Detection (SSD), is known for higher efficiency (Bochkovskiy *et al.*, 2020) and is able to infer both the objects' classes and the bounding boxes simultaneously in one shot.

In this work, the goal is to spatially detect and classify cones from track images, in 4 possible classes: yellow, blue, small orange and large orange. Although accuracy is very important, efficiency was prioritized as the detector should work in a real-time situation, with the onboard hardware limitations. Thus, this work focuses on YOLO-based architectures (Redmon, 2016), known for their efficiency in generic object detection based on CNNs, described in Section 2.4

2.3 Performance Metrics

The *Intersection over Union* (IoU) is a similarity measure used to evaluate how well the predicted bounding box overlaps with the target bounding box in object detection (Fig. 2). From this IoU, one can compute the true positives (*tp*) (no. of objects with $\text{IoU} \geq \gamma$), false positives (*fp*) (no. of objects with $\text{IoU} < \gamma$), and false negatives (*fn*) (no. of undetected objects) and, thus, subsequently the resulting precision ($tp/(tp + fp)$) and recall ($tp/(tp + fn)$) can also be computed. Here, γ represents a threshold for IoU at which objects are considered either a *tp* or *fp*. For some fixed threshold γ , precision measures the fraction of predicted bounding boxes that are actual objects, whereas recall gives the rate of objects that were actually detected (Rezatofighi *et al.*, 2019).

To evaluate the object detectors, the *Average Precision* (AP) metric is used, defined as the area under the precision-recall curve. It can be computed by averaging precision at eleven discrete recall values from 0.0 to 1.1 with a step size of 0.1. In the case of object detection and in this work, the precision is averaged for γ from 0.5 to 0.95 with a step size of 0.05 instead. For multiple classes of objects, the *mean Average Precision* (mAP) is computed which takes the mean of the AP for all classes.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


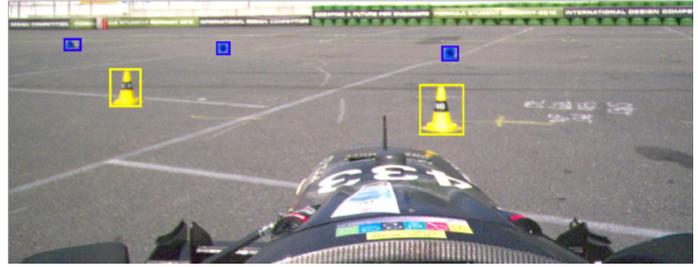


Figure 2: Left: Intersection over Union (IoU) metric for a bounding box class prediction. Right: a typical image from the race track delimited by cones with their associated bounding boxes.

2.4 YOLO

You Only Look Once, or YOLO, is an object detection method (Redmon, 2016) based on CNNs that combines classification and spatial localization in a single pass through the image, making YOLO an exceptionally fast and accurate method. This is achieved by stating the learning problem as a regression problem for finding both the bounding boxes and classes probabilities simultaneously. This means that, differently from other approaches such as *sliding window* algorithms (Goodfellow *et al.*, 2016) and *region proposal-based* (Ren *et al.*, 2015), YOLO processes the whole image in one go, which allows it to implicitly codify contextual spatial information about objects and their classes.

Originally, YOLO (version 1) uses a CNN with 24 layers, where the first 20 convolutional layers from Figure 3 are pretrained on the ImageNet dataset, and are followed by an average-pooling layer and a fully connected layer. The final layer predicts both class probabilities and bounding box coordinates. The width and height of the bounding box are normalized by the image width and height so that its value falls between 0 and 1. The bounding box center x and y coordinates are parametrized to be offsets of a particular grid cell location so they are also bounded between 0 and 1. The activation function for the final layer is the *leaky rectified linear activation*.

The loss function for training YOLO (Redmon, 2016) is formed by sum-squared errors of the predicted values of the bounding boxes as well as of the class probabilities. To reflect that small deviations in large boxes matter less than in small boxes, the model predict the square root of the bounding box width and height instead of the width and height directly (Redmon, 2016).

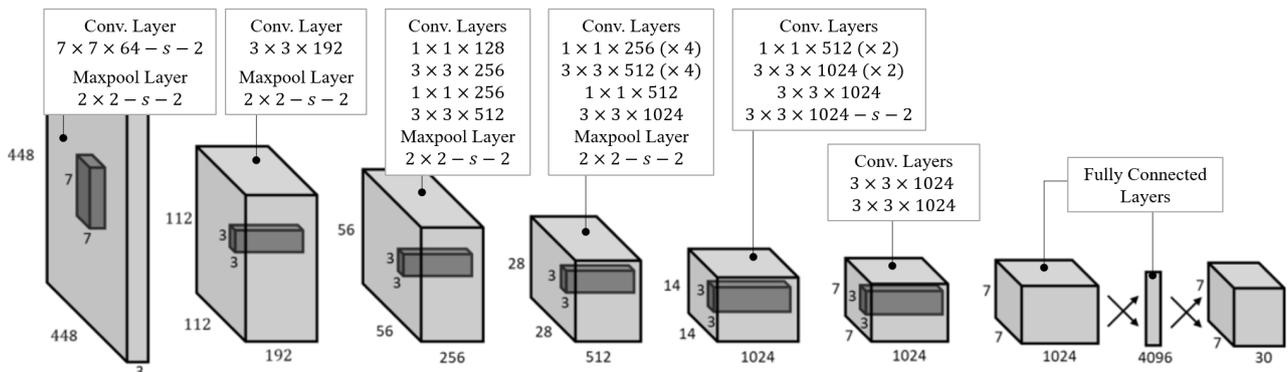


Figure 3: YOLO architecture.

After the first publication of the YOLO network, other versions were released with optimizations that improved performance, and where the whole architecture is composed of three high-level modules, namely, backbone, neck and head, where each one includes many convolution layers. YOLOv3 (Redmon and Farhadi, 2018) had changed the loss function to *cross entropy*, the backbone changed from Darknet-19 to Darknet-53 (Redmon, 2013–2016), predictions were made in three different scales and *residual blocks* were added to the network.

YOLOv4 (Bochkovskiy *et al.*, 2020) uses CSPDarknet53 (Wang *et al.*, 2019) for the backbone, *Spatial Pyramid Pooling* (He *et al.*, 2014) and *Path Aggregation Network* (Liu *et al.*, 2018) for the neck, and YOLOv3 (Redmon and Farhadi, 2018) for the head. Besides that, the network combines a series of optimizations to improve performance: *Weighted-Residual-Connections (WRC)*, *Cross-Stage-Partial-connections (CSP)*, *Cross mini-Batch Normalization (CmBN)*, *Self-adversarial-training (SAT)* and *Mish-activation*, *Mosaic data augmentation*, *CmBN*, *DropBlock regularization*, and *CIoU loss*. Therefore, this model presents an *Average Precision (AP)* significantly higher than its previous versions.

2.5 Dataset for Cone detection

In this work, we use the FSOCO (Formula Student Objects in Context) dataset (Dodel *et al.*, 2020), which segregates cone classes such as blue cone, yellow cone, small orange cone and big orange cone. While it is not an open source dataset, to have access to the dataset, it is necessary to contribute with labeled data. The contribution was made as part of this work and it gives a lifelong access for Ampera Racing Team, allowing the development of future works.

The dataset FSOCO is constantly updated as more teams contribute with labeled images. When this work was developed, it was used approximately 2000 images. For every image in the dataset, there is an associated file that contains the information on the localization of the cones and their corresponding classes. When extra data augmentation was used, the resulting dataset increased 5x in size.

As classes in the dataset are not well balanced, there are many more examples for yellow and blue cones than for orange and big orange cones (Figure 4). To train and test networks, the dataset was divided as follows: 70% for training, 10% for validation, and 20% for testing.

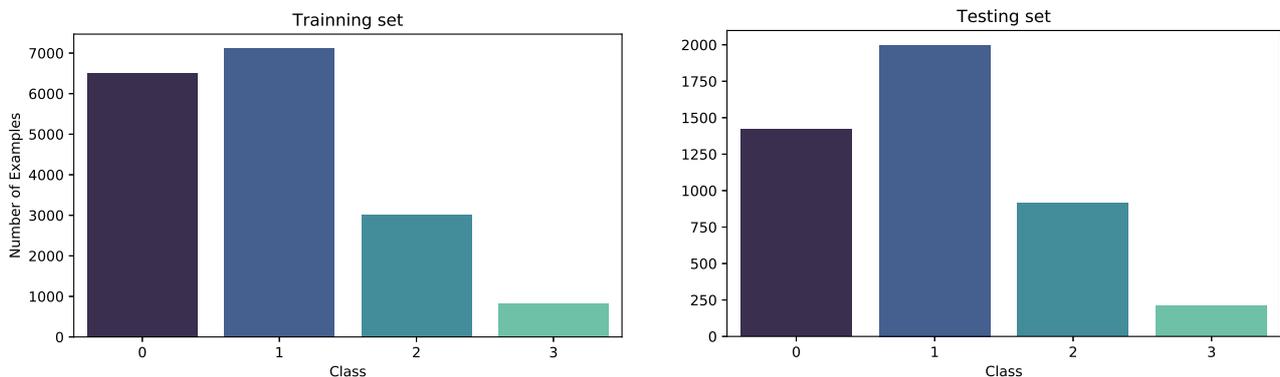


Figure 4: Distribution of cone classes for the training and test sets, for the FSOCO dataset, where 0,1,2 and 3 represent blue, yellow, small orange and big orange cones, respectively.

3. EXPERIMENTAL RESULTS

3.1 Experimental Setup

For all experiments in this work, the following configuration is used, unless otherwise stated:

- **Hardware specification:** CPU Intel(R) Xeon(R) CPU @ 2.00GHz, GPU Tesla P100-PCIE-16GB, 68GB Disk, 12.7 GB RAM and CUDA Version 10.2.
- With YOLO, **transfer learning** is employed by initialization of network weights that were pretrained on the MS COCO dataset, before final training with the dataset of traffic cones.
- **Training parameters:** batch size = 64, subdivisions = 16 and decay = 0.0005. For **YOLOv3**: learning rate = 0.001, max batches = 10000. **YOLOv4**: learning rate = 0.0013, max batches = 10000. **YOLOv4 Tiny**: learning rate = 0.00261, max batches = 6000.
- The **architecture** of the CNNs are the default ones from the respective versions of YOLO (version 3 or 4). They can be visualized here: github.com/laizaparizotto/Cone-Detection-for-Formula-Student.

3.2 Overall Comparison Between Networks

Formula Student competition has a standard track whose boundaries are defined by yellow cones located at the right side of the track and blue cones positioned at the left side. Besides, big orange cones mark the beginning and the end of laps. Therefore, not only the format and color of the cone but also the relative size must be taken into account when performing cone detection. To differentiate well among the 4 possible classes of cones is important so that the racing car has the best conditions to win the race, i.e., is able to perceive the track well in advance as well as to understand when a lap finishes. Results for testing are presented in Table 1, where no frames from the test set are used in training. YOLOv4 has a slightly higher mAP than YOLOv3. While YOLOv3 presented better inference time for cone detection in a whole image of 416x416 (26.8 milliseconds), YOLOv4 performed this operation in 33.2 milliseconds. On the other hand, the tiny version of YOLOv4 drastically reduced the inference time to an average of 5.2 milliseconds at the cost of reducing the detection precision (mAP).

Table 1: Test performance of networks with image input size of 416x416 for FSOCO dataset for IoU threshold of 50%.

	YOLOv3	YOLOv4	YOLOv4 Tiny
Average Inference time (ms)	26.8	33.2	5.2
Mean Average Precision (mAP %)	69.2	76.3	56.8
Blue cone (AP %)	72	79.3	59.5
Yellow cone (AP %)	65.1	72.9	53.1
Orange cone (AP %)	53.4	63.3	38.6
Big orange cone (AP %)	86.5	89.8	78.8

To win the Formula Student Driverless competition, the racing cars should drive as fast as possible, with a minimum number of mistakes. Final score is based on the time the car took to finish the competition and on infringed penalties, such as knocking cones or getting out of track’s boundaries. In this context, one of the most important factors to consider, after being able to detect cones consistently, is the frequency at which the system can run. Thus, reducing the time required for detecting images is crucial to leave enough computation time for the other modules in the pipeline of a racing car, such as control (Kabzan *et al.*, 2019).

From Table 1, one could ask: is it better to use the normal version of YOLOv4 (second column) with higher accuracy, or the Tiny version (third column), with higher speed but less accuracy. To better address this question, in the following section we explore how YOLOv4 Tiny can have its accuracy improved while still operating fast.

3.3 Higher image resolution improves accuracy for YOLOv4 Tiny

As YOLOv4 Tiny operates with a very fast inference time, it is worthy to investigate ways to improve its accuracy while maintaining inference time still low. To achieve that, we train and evaluate networks with higher resolution images, that can provide more detailed information about the objects in the scene. We expect that this change can increase the cone detection accuracy for YOLOv4 Tiny.

Figure 6 presents results with other two higher image resolutions, 512x512 and 608x608, in addition to the standard 416x416. Note that the YOLOv4 Tiny net was trained from scratch for each new chosen resolution. The figure shows the Mean Average precision and inference speed for each configuration. It is impressive that YOLOv4 tiny (608x608) gets as good accuracy as YOLOv3 (416x416), but at a fraction of the inference time.

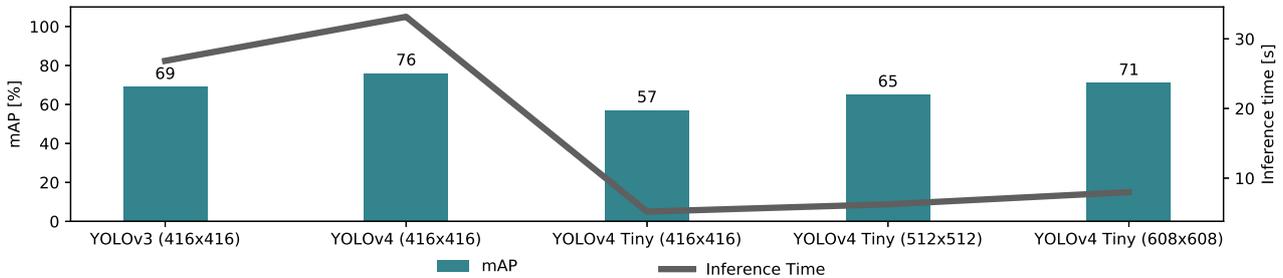


Figure 5: mAP and inference time for the FSOCO dataset.

These results show that by increasing the image input size (resolution), it is possible to improve the accuracy of the network while still maintaining a fast cone detection time: the mAP has raised significantly from 56.8% to 70.8% when switching from 416x416 to 608x608 image resolution (a 24.6% increase in accuracy), whereas the inference time went from 5.2 milliseconds to 8.0 milliseconds, which is still only 29% of the YOLOv3 inference time (26.8 ms).

In Figure 6, the Average Precision was plotted for each individual cone type. It is possible to see that for large orange cones the network presents a higher precision than for other cone types, even with less training examples belonging to that class. This is due to the size that these cones occupy in the scene in relation to the total image size. Their bigger size offers much more information for a reliable detection much in the same way increasing a image resolution provides better accuracy as already observed.

3.4 Error visualization throughout a test image

To understand how the networks considered in this work perform on an usual scene of a racing track with multiple cones in different positions and distances, we propose to color-code the IoU metric for each cone in the image. This can be seen in Figure 7, where a characteristic picture from the test dataset was analyzed. The metric intersection over union (IoU) between the ground true bounding boxes and the network predictions is color-coded in different shades of green. A

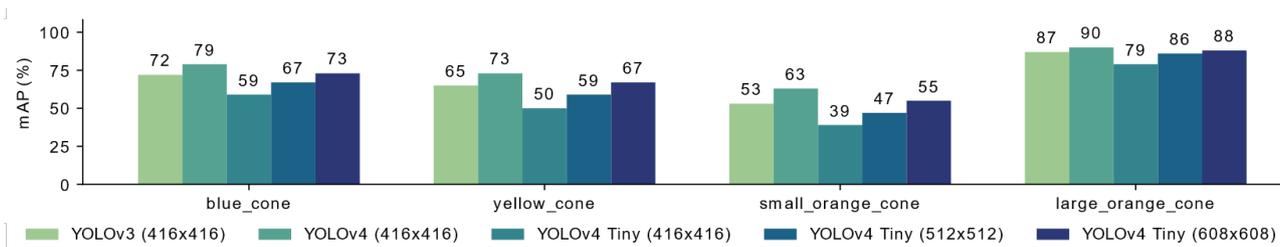


Figure 6: mAP for each class in the FSO CO dataset.

IoU closer to 1 means perfect matching with the true bounding box of the respective cone, and is coded as bright green (rectangle), whereas lower values are coded with darker green. Cones that are not detected appear as dark red rectangles.

It can be seen that YOLOv4 has the most accurate predictions as it has the highest IoU in the predicted bounding boxes. For this image, the standard YOLO has more bright greenish rectangles than the tiny version, especially for closer (bigger) cones, representing a better adjustment of the predicted bounding boxes to the ground truth. Besides, YOLOv4 Tiny gets more accurate as its image input size (resolution) increases. With the 416×416 configuration, the network made wrong predictions for several cones, while the 608×608 configuration only predicted one wrong example. In general, we observe that most of the undetected cones are very small in size, independent of the configuration used, and they do not repeat for each different network.

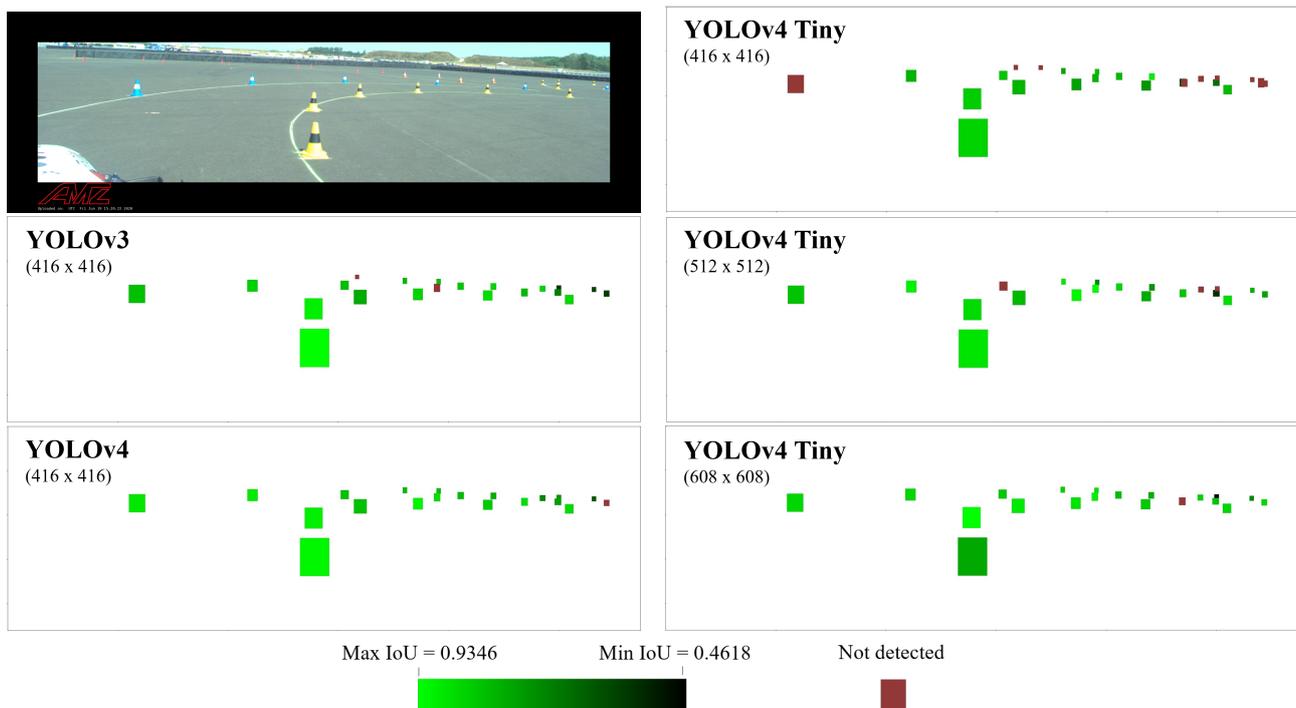


Figure 7: Error visualization. The color gradient is related to the IoU between the ground truth label and the predicted bounding boxes. Dark red rectangles are related to cones that were not predicted by the network.

3.5 Performance under Special Image Conditions

The real-world environment of an autonomous car can be very variable and not always able to provide perfect image conditions. The special image conditions (or perturbations) could be caused by different weather conditions, darkness, occlusion, etc. In this section, the performance of three networks are analyzed – YOLOv4 (416×416), YOLOv4 Tiny (608×608) and YOLOv4 Tiny (608×608) with extra data augmentation – under 4 different conditions: brightness, exposure, blur and noise. Besides, images with manual perturbation, such as the presence of other objects and occlusion, are considered.

Data Augmentation is a regularization technique used to avoid overfitting when training machine learning models. YOLOv4 Tiny (608×608) with extra data augmentation was trained with all images of the FSO CO dataset plus new perturbed images considering all of the following four effects – brightness, exposure, blur and noise – on all images. This means that the dataset was 5 times bigger than the original version. It is important to consider that the YOLOv4

architecture already internally includes several augmentation techniques. Even though, we performed this additional data augmentation on top of the already existing one in YOLOv4 in order to compare results with a standard YOLOv4 Tiny without this extra augmentation.

3.5.1 Brightness, Exposure, Blur and Noise

The first perturbation considered was brightness/darkness, where the parameters vary from +65% (brightness) to -65% (darkness). The second one considers a variation in the exposure condition, from -40% to +40%. The third transformation applies a Gaussian Blur perturbation on images. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen. In physical situations, blur could appear on images when there is a moving camera detecting stationary objects, or it could even simulate a raindrop in the camera. Finally, a salt-and-pepper noise – randomly change some pixels to white or black – was applied. Noise means altering pixels to be totally different from the context of the neighboring pixels, erasing any prior information. In contrast to the humans’ perception system, where noise can be easily ignored, machine learning algorithms usually have a hard time classifying images with noise. Noise can simulate a situation where the camera connection is not stable or even a situation where there is dirt in the camera.

Table 2: Test performance of networks on different image conditions for the FSOCO dataset. Percentages in parenthesis give the relative change of the third column (with extra augmentation) with respect to the second column (without) for IoU threshold of 50%.

	YOLOv4 (416×416)	YOLOv4 Tiny (608×608)	YOLOv4 Tiny w/ aug (608×608)
No perturbation	76.3	70.8	-
Brightness (mAP %)	67.7	61.4	63.7 (+3.7%)
Exposure (mAP %)	75.4	68.9	67.2 (-2.4%)
Blur(mAP %)	69.8	65.0	68.0 (+4.6%)
Noise (mAP %)	67.2	52.3	67.0 (+28.8%)

Table 2 shows the mAP on test images that underwent the application of the respective perturbation or special image condition. Note that YOLOv4 with augmentation was trained including additional perturbed images with all four effects in the training set. The application of any of the effects diminishes the performance of all networks in relation to *No perturbation* (bold line in the table). In general, the non-tiny version of YOLOv4 had a more robust cone detection under image perturbations. On the other hand, the extra data augmentation of the tiny YOLO (third column) served well the purpose of improving detection performance under these conditions. Furthermore, it can be seen that the networks are considerably affected by brightness changes. YOLOv4 Tiny with augmentation presented a slight improvement in its mAP (63.7%) when compared to the traditional YOLOv4 Tiny (61.4%).

With respect to exposure, the networks did not show a substantial decrease in performance. One possible explanation is that the perturbation (+40%) was not as high as it was for brightness (+65%). The decision of only applying +40% was based on the visual change of the images, such that a human is still able to discriminate the cone’s positions and their classes. The only case where data augmentation did not improve the performance was for the exposure effect, which by itself did not affect strongly any of the networks.

The test results for the blur condition are similar to ones for brightness, with a decrease in performance under the special condition and with augmentation recovering partially the lost performance of the tiny YOLO. Finally, for noise perturbation, the extra augmentation process played an important role in improving the YOLOv4 Tiny performance, since the mAP went from 52.3% to 67%, a 28.8% increase.

3.5.2 Out-of-distribution perturbations: different object classes and occlusion

The goal here is to view how the networks perform under non-common situations that can be considered out-of-distribution cases not ever seen in the training set, such as unusual objects in the track, occlusion and other situations that could confuse the detection. This stress testing is an important task because the perception pipeline, which includes cone detection, plays a crucial role in the vehicle’s decision-making.

The first perturbations correspond to including animals in the track, more specifically cats and birds. Some of the cats were cropped in a cone shape in order to challenge the cone detection capability. The second perturbation corresponds to the occlusion of random parts of an image through the drawing of black holes (circles). Figure 8 shows the predictions for these images for the following networks, respectively: YOLOv4 (416 × 416), YOLOv4 Tiny (608 × 608) and YOLOv4 Tiny with Augmentation (608 × 608).

It can be seen that YOLOv4 (416 × 416) is not able to differentiate objects that are very different from cones but slightly resembles the cone shape and size. It detects two birds as a blue cone and two cropped cats as orange cones. On

the other hand, it performs well in the occluded images, being able to correctly detect all of the occluded cones.

The birds were not wrongfully detected as cones by YOLOv4 Tiny (608 × 608), but the cat in the right image was still detected as an orange cone. Besides, both the tiny YOLO networks with higher resolution seem to suffer from occluded images, since they were not able to detect the two yellow cones in the left image and one yellow cone (almost completely covered) in the right image.

Finally, YOLOv4 Tiny with extra Augmentation (608 × 608) deals better with the unusual animals in the track, being able to ignore all of them. As for the occlusions, it had a better confidence score in the blue cone in the middle of the left image, although it was not able to detect the two yellow cones nor the occluded yellow cone in the right image.

We can conclude that augmenting the original dataset with images under the four aforementioned special conditions yields a more robust cone detection method that can ignore out-of-distribution objects that resemble cones in shape. Note that the network was not trained explicitly to ignore these unusual objects and, thus, this capability was obtained indirectly from the augmentation with other types of image perturbations.



Figure 8: YOLOv4 (416x416), YOLOv4 Tiny (608x608) and YOLOv4 Tiny with Augmentation (608x608) detections under out-of-distribution perturbations.

4. CONCLUSION

This work has presented an investigation of the application of different YOLO-based convolutional neural networks for cone detection in images of race tracks from the Formula Student Driverless competition. Experimental results have shown that it is possible to achieve a fast inference time while maintaining very good detection accuracy (mAP) for the four different classes of cones when a tiny version of the YOLO net and a higher image resolution of 608x608 pixels are employed. This result is important in the context of real-time computation for autonomous race cars.

We have shown that image perturbations impact the detection accuracy of version 4 of YOLO, even though the latter already implements regularization techniques such as data augmentation. On the other hand, by explicitly augmenting the training set with perturbed images, we obtained improved mean average precision for test images under these types of perturbations when compared to the YOLO net trained on the original set. Furthermore, this extra data augmentation was shown to be crucial when the test images were altered with out-of-distribution perturbations, such as new object classes (birds and cats) of shape, size, and color similar to ones found in cones.

As future work, the architecture of the CNNs in YOLO could be further investigated in terms of number of convolutional and pooling layers as well as size of the kernel for each layer. This could further enhance the cone detection for more difficult cases: when the cone is very small or occluded, for instance. Integration to the pipeline of a racing car and

usage of the investigation of version 5 of YOLO is currently being done by our group.

5. REFERENCES

- Aggarwal, C.C., 2018. *Neural Networks and Deep Learning*. Springer. ISBN 9783319944623.
- Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M., 2020. “Yolov4: Optimal speed and accuracy of object detection”. *arXiv preprint arXiv:2004.10934*.
- Caporale, D., Settini, A., Massa, F., Amerotti, F., Corti, A., Fagiolini, A., Guiggiani, M., Bicchi, A. and Pallottino, L., 2019. “Towards the design of robotic drivers for full-scale self-driving racing cars”. In *ICRA 2019*, pp. 5643–5649.
- Chen, T., Li, Z., He, Y., Xu, Z., Yan, Z. and Li, H., 2019. “From perception to control: an autonomous driving system for a formula student driverless car”. *CoRR*, Vol. abs/1909.00119.
- de la Iglesia Valls, M., Hendrikx, H.F.C., Reijgwart, V., Meier, F.V., Sa, I., Dubé, R., Gawel, A.R., Bürki, M. and Siegwart, R., 2018. “Design of an autonomous racecar: Perception, state estimation and system integration”. *CoRR*, Vol. abs/1804.03252.
- Dhall, A., Dai, D. and Gool, L.V., 2019. “Real-time 3d traffic cone detection for autonomous driving”. *2019 IEEE Intelligent Vehicles Symposium (IV)*.
- Dodel, D., Schötz, M. and Vödisch, N., 2020. “FSOCO: the formula student objects in context dataset”. *CoRR*, Vol. abs/2012.07139.
- Dodge, S.F. and Karam, L.J., 2016. “Understanding how image quality affects deep neural networks”. *CoRR*, Vol. abs/1604.04004.
- Goodfellow, I.J., Bengio, Y. and Courville, A., 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA.
- He, K., Zhang, X., Ren, S. and Sun, J., 2014. “Spatial pyramid pooling in deep convolutional networks for visual recognition”.
- Kabzan, J., Valls, M.d.l.I., Reijgwart, V., Hendrikx, H.F.C., Ehmke, C., Prajapat, M., Bühler, A., Gosala, N., Gupta, M., Sivanesan, R. *et al.*, 2019. “Amz driverless: The full autonomous racing system”. *arXiv preprint arXiv:1905.05150*.
- Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. and Pietikäinen, M., 2019. “Deep learning for generic object detection: A survey”. *International Journal of Computer Vision*, Vol. 128, No. 2, p. 261–318.
- Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J., 2018. “Path aggregation network for instance segmentation”. In *2018 IEEE/CVF*. pp. 8759–8768.
- Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A.S., Bethge, M. and Brendel, W., 2019. “Benchmarking robustness in object detection: Autonomous driving when winter is coming”. *CoRR*, Vol. abs/1907.07484.
- Redmon, J., 2013–2016. “Darknet: Open source neural networks in c”.
- Redmon, J. and Farhadi, A., 2018. “Yolov3: An incremental improvement”. *arXiv preprint arXiv:1804.02767*.
- Redmon, J.S.D.R.G.A.F., 2016. “(YOLO) You Only Look Once”. *Cvpr*. ISSN 01689002.
- Ren, S., He, K., Girshick, R.B. and Sun, J., 2015. “Faster R-CNN: towards real-time object detection with region proposal networks”. *CoRR*, Vol. abs/1506.01497.
- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S., 2019. “Generalized intersection over union: A metric and a loss for bounding box regression”. In *CVPR*.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L., 2015. “ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, pp. 211–252.
- Shafiee, M.J., Jeddi, A., Nazemi, A., Fieguth, P. and Wong, A., 2021. “Deep neural network perception models and robust autonomous driving systems: Practical solutions for mitigation and improvement”. *IEEE Signal Processing Magazine*, Vol. 38, No. 1, pp. 22–30.
- Simonyan, K. and Zisserman, A., 2015. “Very deep convolutional networks for large-scale image recognition”.
- Strobel, K., Zhu, S., Chang, R. and Koppula, S., 2020. “Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions”.
- Tian, H., Ni, J. and Hu, J., 2018. “Autonomous driving system design for formula student driverless racecar”. *CoRR*, Vol. abs/1809.07636.
- Wang, C., Liao, H.M., Yeh, I., Wu, Y., Chen, P. and Hsieh, J., 2019. “Cspnet: A new backbone that can enhance learning capability of CNN”.
- Wu, Y., Kirillov, A., Massa, F., Lo, W.Y. and Girshick, R., 2019. “Detectron2”.

6. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.