xYOLO: A Model For Real-Time Object Detection In Humanoid Soccer On Low-End Hardware

Daniel Barry *, Munir Shah ¶, Merel Keijsers †,
Humayun Khan § and Banon Hopman ‡

Departments: * *Computer Science* † § *HIT Lab NZ* ‡ *School of Product Design*
Emails: * dan.barry@pg.canterbury.ac.nz ¶ munirsha@gmail.com
† merel.keijsers@pg.canterbury.ac.nz ‡ bmh77@uclive.ac.nz § humayun.khan@pg.canterbury.ac.nz

University of Canterbury
Christchurch, New Zealand

Abstract—With the emergence of onboard vision processing for areas such as the internet of things (IoT), edge computing and autonomous robots, there is increasing demand for computationally efficient convolutional neural network (CNN) models to perform real-time object detection on resource constraints hardware devices. Tiny-YOLO is generally considered as one of the faster object detectors for low-end devices and is the basis for our work. Our experiments on this network have shown that Tiny-YOLO can achieve 0.14 frames per second (FPS) on the Raspberry Pi 3 B, which is too slow for soccer playing autonomous humanoid robots detecting goal and ball objects. In this paper we propose an adaptation to the YOLO CNN model named xYOLO, that can achieve object detection at a speed of 9.66 FPS on the Raspberry Pi 3 B. This is achieved by trading an acceptable amount of accuracy, making the network approximately 70 times faster than Tiny-YOLO. Greater inference speed-ups were also achieved on a desktop CPU and GPU. Additionally we contribute an annotated Darknet dataset for goal and ball detection.

Index Terms—CNN, RoboCup, Object Detection

I. INTRODUCTION

The popularization of deep learning has done much to further advancements in computer vision, where modest amounts of computational power allow for the processing of images to gain insight on their contents [11]. As real-world image data typically has high spatial correlation, convolutional neural networks (CNNs) have been particularly successful in the application of object detection in images [26]. Compared to fully connected networks, CNNs offered large computation and size reduction [6].

In this paper our chosen problem domain is RoboCup, an annually held international humanoid robotics competition with the goal of producing a team of robots that beat the best human football players in the world by 2050 [9]. The reason for this ambitious goal is to help motivate several key areas in artificial intelligence in a format that the general public understand and appreciate without having insight to the complexity of the robots themselves. There are many limitations in the competition in aid of reaching the 2050 goal, such as: robots must be fully autonomous, must be human like (e.g. passive sensors only) and must adhere to an adaptation of the official FIFA rules [1].

Our team, *Electric Sheep*, competed in the 2019 world cup with our unique low-cost open-source humanoid robotics platform [2]. The purpose of designing and building a low-cost platform was to lower the boundary to entry, as larger robots have seen larger costs in recent years which could discourage new teams from entering the competition. Our platform, *Black Sheep*, performs all processing on a Raspberry Pi 3 B including our vision processing pipeline. As our agent behaviour is very simple it only requires the detection of goals and balls.

In this paper, we propose a YOLO based CNN model which can detect balls and goal posts at ≈10 FPS, which given the current relatively slow speed of robot play, is an acceptable frame rate. Our proposed model is called xYOLO and exploits the domain specific attributes such as the requirement of two classes (ball and goal post), simple shape features of the objects and clearly differentiable objects from the background. This allows our model to achieve real-time object detection speed with reasonable accuracy.

In Section II we give a brief overview of efforts prior to the implementation of the neural networks in this domain, why these approaches are not appropriate for the updated environment and the current approaches for object detection. In Section III we describe our network architecture for xYOLO and how it differs from similar existing work. For Section IV we describe experiments and analyse results for comparable networks. Finally, in Section V we evaluate our work and discuss future work.

II. RELATED WORK

Traditionally, in the context of the RoboCup humanoid robotics competition, colour segmentation based techniques have been used to detect features of the soccer field, such

[1]Humanoid rules: https://www.robocuphumanoid.org/materials/rules/
[2]Humanoid platform: https://github.com/electric-sheep-uc/black-sheep

as goals and balls [16] [17]. These techniques are fast and can achieve good accuracy in simplistic environments, for example the use of an orange ball, controlled indoor lighting and yellow coloured goals. However, in light of RoboCup's 2050 goal, teams have seen the introduction of natural lighting conditions (exposure to sunlight), white goal posts with white backgrounds and FIFA balls with a variety of colours. Colour segmentation based techniques fail to perform in these challenging scenarios and has mostly pushed the competition towards implementing a variety of neural network approaches [12] [25].

CNN based models have shown great progress in terms of object detection accuracy in complex scenarios [21] [15], [10] [24] [20] [1]. However, these high performing computer vision systems based on CNNs, although much leaner than fully connected networks, are still both considerably memory and computationally exhausting, and achieve real-time performance only on high-end GPU devices. For this reason, most of these models are not suitable for low-end devices such as smart phones or mobile robots. This limits their use in real-time applications such as autonomous humanoid robots playing soccer, as there are power and weight considerations. Thus, the development of lightweight, computationally efficient models that allow CNNs to work using less memory and on minimal computational resources is an active research area [3] [1] [5] [23].

Recently, a large number of research papers have been published on the topic of lightweight deep learning models for object detection that are suitable for low-end hardware devices [3] [22] [2] [13] [18] [21] [5] [23]. Most of these models are based on SSD [15], SqueezeNet [7], AlexNet [10], and GoogLeNet [24]. Generally, in these models the object detection pipeline contains several components such as pre-processing, large numbers of convolution layers, and post-processing. Classifiers are evaluated at various locations in images and at multiple scales using a sliding window approach or region proposal methods. These complex object detection pipelines are computationally intensive and consequently slow. XNOR-Networks [18] approximate convolutions using binary operation, which is computationally efficient compared to the floats used in traditional convolutions. An obvious downside of XNOR networks is the reduction in accuracy for similarly sized networks.

On the other-hand, in *you only look once* (YOLO), object detection is framed as a single regression problem. YOLO works at the bounding box level rather than pixel level, i.e. YOLO simultaneously predicts bounding boxes and associated class probabilities from the entire image in one "look". One of the key advantages of YOLO is its ability to encode contextual information, and as a result it makes less mistakes in confusing background patches in an image for objects [20] [21].

The "lighter" version of YOLO v3 [21], called Tiny-YOLO, was designed with speed in mind and is generally reported as one of the better performing models in-terms of speed and accuracy trade-off [21]. Tiny-YOLO has nine convolutional layers and two fully connected layers. Our experiments suggest that Tiny-YOLO is able to achieve 0.14 FPS on Raspberry Pi 3, which is far from real-time object detection.

From the results reported in [21], it can be concluded that these object detectors are not able to give real-time performance on low-end hardware with minimal computing resources (e.g. humanoid robots with a Raspberry Pi as the computing resource). In our robots, we are using one compute resource for several different processes, such as the walk engine, self-localization, etc. The vision system is left with approximately a single core to perform all object detection.

III. Network

Our proposed network, xYOLO, is derived from *YOLO v3 tiny* [21], specifically we use AlexeyAB's Darknet fork that allows for XNOR layers and building on the Raspberry Pi [3]. As seen in Figure 1, xYOLO utilizes both normal convolutional and XNOR layers in both training and recall. The network has several key changes:

- *Reduction in input layer size:* Scaling the input image to 256 x 256 pixels was the smallest input we could create without sacrificing the network's ability to see details at far-distance in the 640 x 480 original image. Due to limitations of the framework implementation, preserving aspect ratio was not easily possible. Switching from RGB to grey scale input (three channels down to one) had very little impact on speed, but largely affected detection quality, hence we use full colour information. Through experimentation we realized that ball detection relies on its context, in this case the green field background.

- *Heavily reduced number of filters:* Generally, the objects we are attempting to detect are quite simple in shape and features, meaning this domain specific reduction can be made. We were able to heavily reduce the size of the network with this change and increase detection speed dramatically.

- *Layers m, n, o and p use XNOR:* Through experimentation we found that this part of the network was able to switch to XNOR without affecting training or prediction. Whilst the network size remains the same, not using floating point arithmetic gave a marginal speed increase during detection. When using XNOR throughout the network (specifically the convolutional layers between a and k) we found the network was unable to detect objects to any accuracy (see Figure 2). We believe the early feature formation in the network to be highly important in training and object detection for small networks.

Each year of the RoboCup competition introduces new challenges, where models have to be retrained using images collected and labelled during the setup time of the competition. Consequently, our approach towards designing this network was to reduce the training time to below 45 minutes, allowing for relatively rapid testing of different network configurations and new soccer field conditions. Figure 2 is an example of a network where the parameters are reduced too far, such that

---

[3]Darknet fork: https://github.com/AlexeyAB/darknet
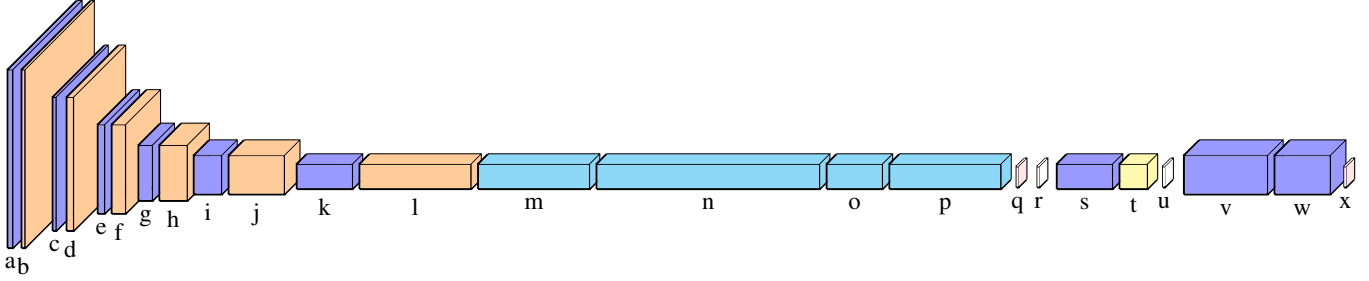
Fig. 1: The network structure is as follows: convolutional: a, c, e, g, i, k, s, v, w, max pool: b, d, f, h, j, l, convolutional XNOR: m, n, o, p, yolo: q, x, upsample: t, route: r, u.
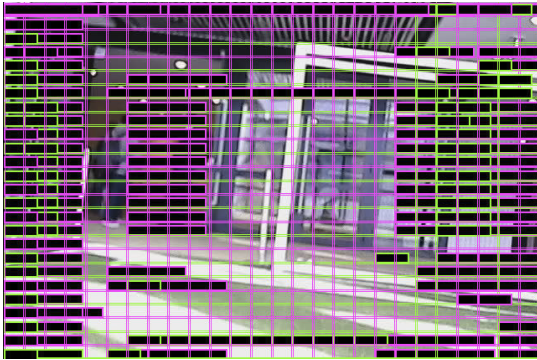

Fig. 2: Example of a network that is too small to detect objects.

it is incapable of detecting objects. In Figure 3 this would manifest itself as the loss mean square error not reducing below 6 before 1,000 iterations or models that are not able to reduce their loss to an acceptable value, i.e. below 1.5. Generally we are able to conclude whether a network has a reasonable chance of success within the first 15 minutes of training.

IV. EXPERIMENTS AND RESULTS

Experiments are conducted using Darknet [19], an open source neural network framework. Darknet is fast (written in C language with many optimizations), easy to compile on the Raspberry Pi and supports both CPU and GPU training and detection. Our proposed model (xYOLO) is compared against Tiny-YOLO (v3) [21] and Tiny-YOLO-XNOR (v3) [18]. Tiny-YOLO is reported as a CNN model with good trade-off between computational efficiency and object detection accuracy [13] [8]. Tiny-YOLO-XNOR is a lightweight implementation of Tiny-YOLO with XNOR [18] in the Darknet framework. Each of the models is adapted for the use of two classes by adjusting the number of filters before the YOLO layers to the following (we have two classes, *ball* and *goal*):

$$filters = (classes + 5) \times 3 \qquad (1)$$

ID	Filters	Size	Input	Output
a	2	3 x 3 / 1	256 x 256 x 3	256 x 256 x 2
b		2 x 2 / 2	256 x 256 x 2	128 x 128 x 2
c	4	3 x 3 / 1	128 x 128 x 2	128 x 128 x 4
d		2 x 2 / 2	128 x 128 x 4	64 x 64 x 4
e	8	3 x 3 / 1	64 x 64 x 4	64 x 64 x 8
f		2 x 2 / 2	64 x 64 x 8	32 x 32 x 8
g	16	3 x 3 / 1	32 x 32 x 8	32 x 32 x 16
h		2 x 2 / 2	32 x 32 x 16	16 x 16 x 16
i	32	3 x 3 / 1	16 x 16 x 16	16 x 16 x 32
j		2 x 2 / 2	16 x 16 x 32	8 x 8 x 32
k	64	3 x 3 / 1	8 x 8 x 32	8 x 8 x 64
l		2 x 2 / 1	8 x 8 x 64	8 x 8 x 64
m	128	3 x 3 / 1	8 x 8 x 64	8 x 8 x 128
n	32	1 x 1 / 1	8 x 8 x 128	8 x 8 x 32
o	64	3 x 3 / 1	8 x 8 x 32	8 x 8 x 64
p	21	1 x 1 / 1	8 x 8 x 64	8 x 8 x 21
s	16	1 x 1 / 1	8 x 8 x 32	8 x 8 x 16
t		2x	8 x 8 x 16	16 x 16 x 16
v	32	3 x 3 / 1	16 x 16 x 48	16 x 16 x 32
w	21	1 x 1 / 1	16 x 16 x 32	16 x 16 x 21

TABLE I: The following is the network structure of xYOLO.

The models are evaluated on our dataset. The details of the dataset are described in Section IV-A. All models were trained using 90% of images in the dataset and 10% of these images were used during testing. Object detection accuracy is measured by mean Average Precision (mAP) and F-Score [14]. All of these models are evaluated using the default parameters settings. Computational efficiency of the models is measured by inference time and train time (minutes). Models are evaluated on the Raspberry Pi 3 B, a standard desktop CPU (Intel i7-6700HQ) and a standard GPU (Nvidea GTX 960M) environment to measure inference time. Since memory is also an issue on low-end hardware, we also compared models using network size MB (Mega Bytes) and computational size BFLOPs (Billion FLOPs) performance metrics [21].

A. Dataset

One of the contributions of this study is our annotated dataset from the RoboCup 2019 competition using cameras mounted on the robots in both the controlled and natural lighting scenarios. We also used some images from previous

competitions via the Image Tagger community-driven project [4]. Each of these raw images are manually annotated. There are two classes in the dataset: ball and goal post. Traditionally people used complete goals as a single object. The inside of the goal is hollow and usually only part of the goal on the field is in the camera frame, making the detection of a full goal often difficult. In RoboCup, generally the ball stays on the ground, thus the robot rarely needs to look upwards and detecting only the bottom of the goal posts is ideal. In consideration of this, we used bottom of the goal posts to detect goal. In this dataset both left and right goal posts are considered as two instances of the same class (goal post).

This dataset contains range of challenging scenarios, such as natural lighting (sun light spots on the field), shadows, and blurred images since robots are moving. Some of the glimpses of the complexities of the dataset can be seen in Figure 5. We have open sourced this dataset and is available for public use [4].

B. Comparative Computational Speed

The key focus of this paper is to achieve real-time object detection and localization performance on low-end computing hardware such as Raspberry Pi 3 B. To time models training duration, we used a cloud instance with an Nvidea K80 GPU and 55GB RAM. All models were trained for 6000 iterations and tested for inference speed in both a standard desktop environment and a Raspberry Pi. As shown in Table II, train times are reported on the cloud instance GPU and inference speed is reported on multiple hardware targets.

As shown in the Table II, xYOLO achieved superior performance in terms of computational efficiency compared to the other tested models. For train time xYOLO is ≈5 times faster than the other two models. For inference speed, xYOLO achieved 706.36 FPS on the GPU, which is ≈7 times faster than Tiny-YOLO-XNOR and ≈9 times faster than Tiny-YOLO. On desktop CPU, xYOLO performed 87 and 35 times better than Tiny-YOLO and Tiny-YOLO-XNOR respectively. On the Raspberry Pi, xYOLO performed 69 times faster than Tiny-YOLO and 25 times faster than Tiny-YOLO-XNOR. The improved speed gain is due to small input and filters size. Tiny-YOLO-XNOR averaged 2.52 FPS on the Raspberry Pi and Tiny-YOLO at 1 FPS. Both of these models are too slow to be used effectively in the RoboCup competition, as games develop quickly. On the other hand, xYOLO is capable of ≈10 fps on the Raspberry Pi, which is reasonable object detection speed, especially for the purpose of humanoid league soccer matches.

In terms of network size, xYOLO is 45 times smaller than Tiny-YOLO and ≈15 times smaller than Tiny-YOLO-XNOR in network size. Similarly, xYOLO requires 0.039 BFLOPs, which is significantly lower than other two models. In short, xYOLO outperformed other models on all computational efficiency metrics.

Network	Tiny-YOLO	Tiny-YOLO-XNOR	xYOLO
Train Time (minutes)	183	174	**39**
Inference (FPS)	0.89	2.52	70.69
i7-6700HQ stdev	0.059	0.026	0.0018
Inference (FPS)	79.97	97.28	706.36
GTX 960M stdev	0.00064	0.00055	0.00023
Inference (FPS)	0.14	0.39	**9.66**
rPi 3 B stdev	0.064	0.012	0.0012
Size (MB)	34.7	12.46	**0.82**
BFLOPs	5.449	5.449	**0.039**

TABLE II: Computational efficiency and network size results. Note that the reported BFLOPs for Tiny-YOLO-XNOR and xYOLO incorrectly calculate the XNOR layers as FLOPs.
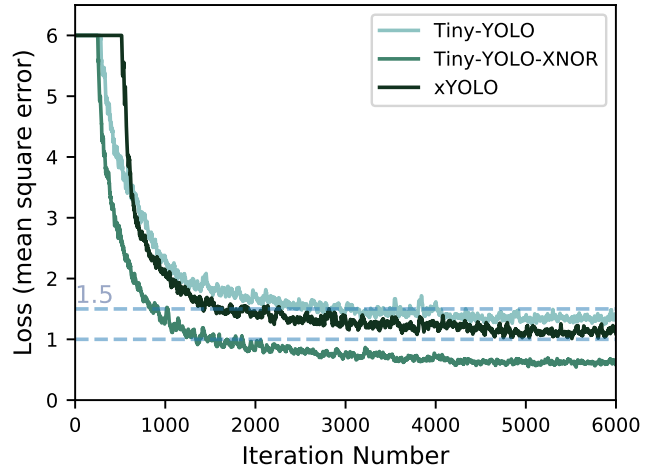

Fig. 3: Train loss results for the models. It is observed that the models took ≈4000 iterations to reach their lowest loss.

C. Comparative Accuracy

The object detection and localization accuracy of the models is measured by train loss (mean square error), validation mAP (mean Average Precision on validation dataset), inference mAP on test dataset and F-score. We used Darknet transfer learning where pre-trained weights (`darknet53.conv.74`) for the Imagenet dataset are used as initial weights for training. This transfer learning helps models take less than 1000 iterations to reduce their loss to less than 6. Figure 3 shows train loss for the models. Results from the Figure 3 suggests that the models took ≈4000 iterations to stabilized and after that loss was not greatly reduced. Tiny-YOLO was able reduce its loss to ≈0.5, with xYOLO to ≈1 and Tiny-YOLO-XNOR to ≈1.5. Accuracy results (mAP) on the validation set are

Algorithm	mAP (Train)	mAP (Test)	F-score
Tiny-YOLO	94.65	93.69	92
Tiny-YOLO-XNOR	68.42	64.75	62
xYOLO	68.22	66.75	68

TABLE III: Object detection accuracy results on our humanoid soccer dataset (see Section IV-A).
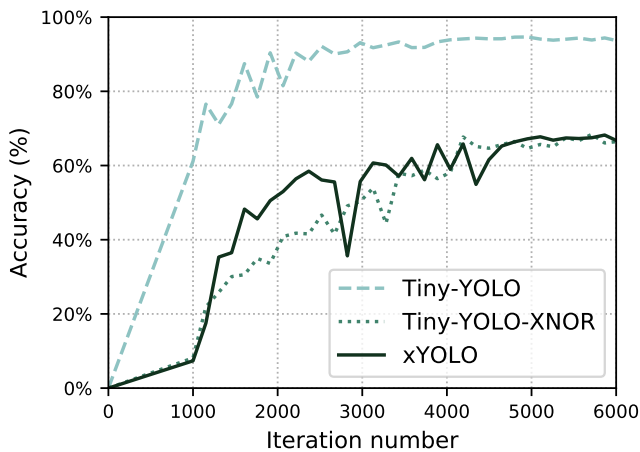
Fig. 4: Accuracy (mAP) results for the models during training on the validation set. It is observed that Tiny-YOLO performed significantly better than the other two models.

presented in Figure 4 and Table III. It is observed that models have achieved similar accuracy on both train and unseen test sets. Tiny-YOLO achieved significantly better object detection accuracy compared to other models. xYOLO was able to achieve ≈68% accuracy on validation dataset, and ≈67% on the test set, which is good when the speed and size of xYOLO is considered.

D. Qualitative Evaluation

Figure 5 demonstrates object detection results by each of the models on a challenging scenarios. All models were able to detect both ball and goal post in easy scenarios, where objects are quite clearly seen. It is observed that the Tiny-YOLO struggled to detect one goal post in the blurred image scenarios (Figure 5b). In a scenario where there are shadows on the field (Figure 5c), Tiny-YOLO-XNOR was not able to detect both goal post or ball, whereas xYOLO was able to detect the ball but not the goal post and Tiny-YOLO was able to detect both objects. In the natural lighting scenario with strong sunlight spots on the field, all models performed well with Tiny-YOLO able to detect partially observable ball. In summary, both Tiny-YOLO and xYOLO have shown advantages in different scenarios.

V. DISCUSSION

Although xYOLO has less accuracy, it is the only model tested that was able to achieve ≈10 FPS with an acceptable ≈70% accuracy, making it suitable for low-end hardware real-time detection on the Raspberry Pi. For humanoid soccer, robots have to make quick decisions (e.g. to detect a rolling ball). For this reason, fast models with slightly lower accuracy work better than highly accurate but slower models. xYOLO provides a good speed and accuracy compromise for humanoid soccer, which was achieved by reducing the training time, thereby reducing experiment time and allowed for us to fine tune the network to detect objects within the domain.

In future work we look towards performing pre-processing techniques on the input image to further reduce the size of the network. Additionally we want to leverage the high correlation of inter-frame data through the use of optical flow.

VI. REFERENCES

[1] Albani, Youssef, Suriani, Daniele Nardi, and Domenico Daniele Bloisi. A Deep Learning Approach for Object Recognition with NAO Soccer Robots. *RoboCup International Symposium*, 2016.
[2] Nicolás Cruz, Kenzo Lobos-Tsunekawa, and Javier Ruiz del Solar. Using Convolutional Neural Networks in Robots with Limited Computational Resources: Detecting NAO Robots while Playing Soccer. *RoboCup International Symposium*, 2017.
[3] Nicols Cruz, Kenzo Lobos-Tsunekawa, and Javier Ruiz del Solar. Using Convolutional Neural Networks in Robots with Limited Computational Resources: Detecting NAO Robots while Playing Soccer. *RoboCup International Symposium*, 2017.
[4] Niklas Fiedler, Marc Bestmann, and Norman Hendrich. ImageTagger: An Open Source Online Platform for Collaborative Image Labeling. In *RoboCup 2018: Robot World Cup XXII*. Springer, 2018.
[5] Alexander Gabel, Tanja Heuer, Ina Schiering, and Reinhard Gerndt. Jetson, where is the Ball? Using Neural Networks for Ball Detection at RoboCup 2017. *RoboCup International Symposium*, 2018.
[6] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jrgen Schmidhuber. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. PhD thesis, Technische Universitat Munchen, 2001.
[7] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size, 2016.
[8] Mohammad Javadi, Sajjad Azar, Azami, Ghidary, Soroush Sadeghnejad, and Jacky Baltes. Humanoid Robot Detection using Deep Learning: A Speed-Accuracy Tradeoff. *RoboCup International Symposium*, 2017.
[9] Kitano, Hiroaki and Asada, Minoru and Kuniyoshi, Yasuo and Noda, Itsuki and Osawa, Elichi. RoboCup: The Robot World Cup Initiative. AGENTS '97, pages 340–347. ACM, 1997.
[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.
[12] Francisco Leiva, Nicols Cruz, Ignacio Bugueo, and Javier Ruiz del Solar. Playing Soccer without Colors in the SPL: A Convolutional Neural Network Approach. *RoboCup International Symposium*, 2018.
[13] Yuxi Li, Jiuwei Li, Weiyao Lin, and Jianguo Li. Tiny-dsod: Lightweight object detection for resource-restricted usages. *arXiv preprint arXiv:1807.11013*, 2018.
[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.
[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science*, page 2137, 2016.
[16] Jacob Menashe, Josh Kelle, Katie Genter, Josiah Hanna, Elad Liebman, Sanmit Narvekar, Ruohan Zhang, and Peter Stone. Fast and Precise Black and White Ball Detection for RoboCup Soccer. *RoboCup International Symposium*, 2017.
[17] Mihai Polceanu, Fabrice Harrouet, and Cdric Buche. Fast Multi-Scale fHOG Feature Extraction Using Histogram Downsampling. *RoboCup International Symposium*, 2018.
[18] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
[19] Joseph Redmon. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/, 2013–2016.
[20] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2015.
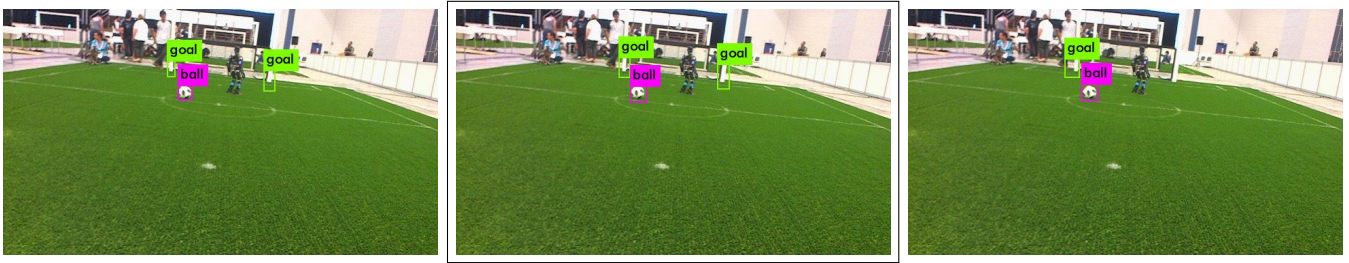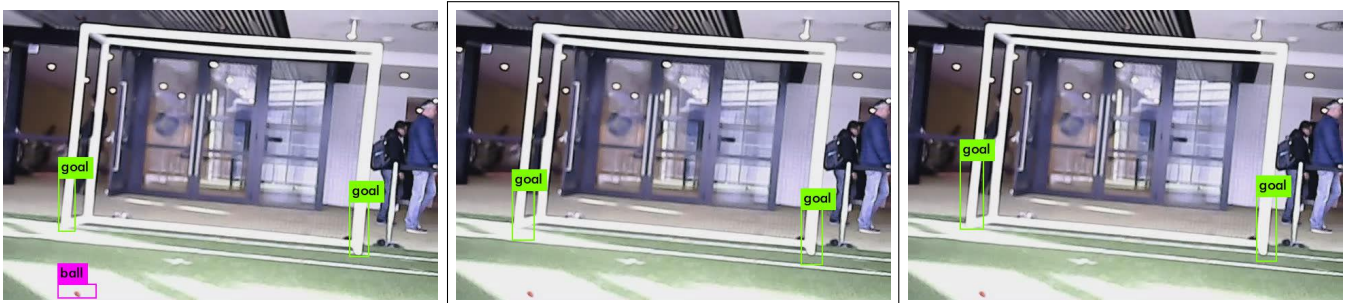
(a) Easy detection scenario, ball and goal posts are clear.
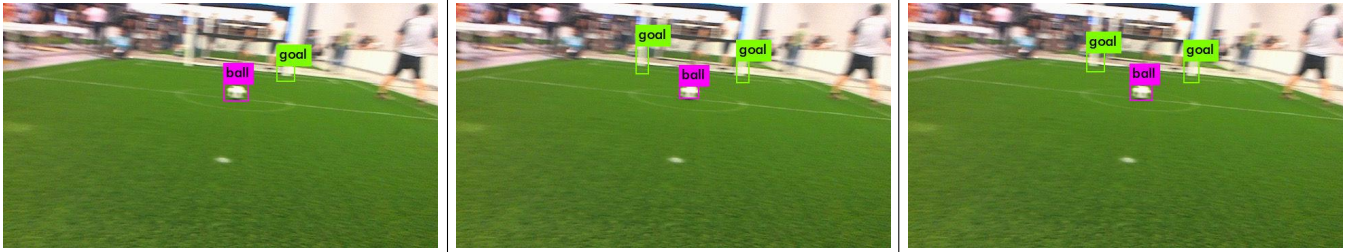
(b) As robot's camera is often moving and as a result models have to perform in blurred image scenario.
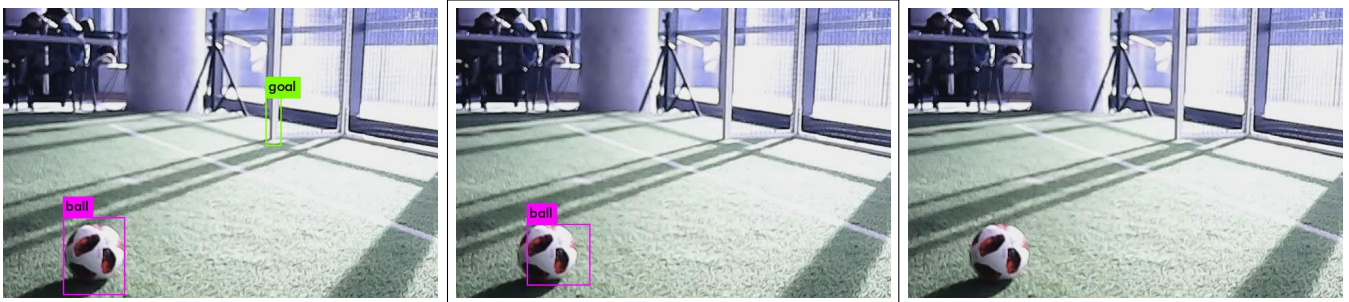
(c) Object detection results in natural lighting scenario with shadows on the field.

(d) Object detection results in natural lighting scenarios with strong sunlight spots on the field.

Fig. 5: Example object detection results produced by the models. Left side: Tiny-YOLO [21], center: xYOLO, right side: Tiny-YOLO-XNOR. Balls and goals are labelled when each network identifies an object that reaches the detection threshold. It can be observed that xYOLO has better object detection results than Tiny-YOLO-XNOR [18] and comparable result to Tiny-YOLO.

[21] Joseph Redmond and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv*, 2018.

[22] Daniel Speck, Pablo Barros, Cornelius Weber, and Stefan Wermter. Ball Localization for Robocup Soccer using Convolutional Neural Networks. *RoboCup International Symposium*, 2016.

[23] Daniel Speck, Marc Bestmann, and Pablo Barros. Towards Real-Time Ball Localization using CNNs. *RoboCup International Symposium*, 2018.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] Sander van Dijk and Marcus Scheunemann. Deep Learning for Semantic Segmentation on Minimal Hardware. *RoboCup International Symposium*, 2018.

[26] Matthew Zeller and Rob Fergus. Visualizing and Understanding Convolutional Networks. *European Conference on Computer Vision*, 2014.