A Real-Time Ball Detection Approach Using Convolutional Neural Networks

Meisam Teimouri [0000-0003-0465-6528] [1], Mohammad Hossein Delavaran [0000-0002-0756-2642] [1] and Mahdi Rezaei [0000-0003-3892-421X] [2]

[1] Mechatronic Research Laboratory, Qazvin Islamic Azad University, Qazvin, Iran
[2] Auckland University of Technology, Auckland, New Zealand

{m.teimouri, mh.delavaran}@qiau.ac.ir, mahdi.rezaei@aut.ac.nz

Abstract. Ball detection is one of the most important tasks in the context of soccer-playing robots. The ball is a small moving object which can be blurred and occluded in many situations. Several neural network based methods with different architectures are proposed to deal with the ball detection. However, they are either neglecting to consider the computationally low resources of humanoid robots or highly depend on manually-tuned heuristic methods to extract the ball candidates. In this paper, we propose a new ball detection method for low-cost humanoid robots that can detect most soccer balls with a high accuracy rate of up to 97.17%. The proposed method is divided into two steps. First, some coarse regions that may contain a full ball are extracted using an iterative method employing an efficient integral image based feature. Then they are fed to a light-weight convolutional neural network to finalize the bounding box of a ball. We have evaluated the proposed approach using a comprehensive dataset and the experimental results show the efficiency of our method.

Keywords: Ball Detection, Convolutional Neural Networks, Humanoid Robot, RoboCup.

1 Introduction

Object detection is the task of classifying meaningful and semantic regions followed by precisely estimating the location of objects in an image [1]. It plays a crucial role in establishing the world model of a soccer-playing robot. One of the most important objects that every soccer-playing robot have to detect is the ball. It is a small moving object that in many situations is occluded by playing robots in the field. Also, ball detection is affected adversely by image blurring caused due to unstable walking of low-cost humanoid robots and slow shutter speed cameras mounted on them. The difficulty of ball detection increases, even more, when the standard size FIFA ball comes with a different pattern in every competition.

With the development of deep neural networks (DNNs), state-of-the-art methods have scored high accuracy in object detection [2, 3]. However, these methods are considered as real-time only by using dedicated graphic processing units (GPUs). In

RoboCup 2018, most of the qualified teams in humanoid soccer leagues have used only a CPU for all kind of tasks, including object detection, walking, world modeling, and behavior analysis. So implementing the DNN-based methods are not feasible on most soccer-playing humanoid robots. Recently several studies have been accomplished to adopt the DNNs for object detection, particularly ball detection on computationally low powered systems. These studies can be divided into two different approaches: single-stage detectors and candidate-based classifiers. Single-stage detectors process a full-sized image using a DNN to localize the position of the interested objects. These methods either lacked a real-time performance on a CPU or missed generalization performance on a real robot. On the other hand, candidate-based classifiers first extract ball candidates precisely and then each candidate is classified as a ball or background class separately. Although these approaches achieved some accurate results on low powered systems, they have highly relied on some manually-tuned heuristic methods to extract the candidates of a specific ball type.

In this paper, we propose a new ball detection method which achieves highly accurate results on a real robot, equipped only with a CPU in real-time performance. The method is described in two steps. In the first step, we find some coarse regions that may contain a ball. In the second step, these regions are fed into a convolutional neural network (CNN) to estimate the bounding box of the ball. The main contributions of this paper are: (i) proposing an iterative algorithm to find regions that probably contain a full ball using an efficient and novel feature extraction technique that is applicable for most soccer balls. (ii) introducing a light-weight CNN that localizes and classifies a ball within the candidate regions of interest. This network can inference the input in real-time on a computationally low powered system.

The rest of the paper is organized as follows: in section 2 we review and investigate recently proposed ball detection methods. Our ball regions detector is presented in section 3. Designing and training the proposed CNN are described in section 4. In section 5 we provide experimental results and then conclude in section 6.

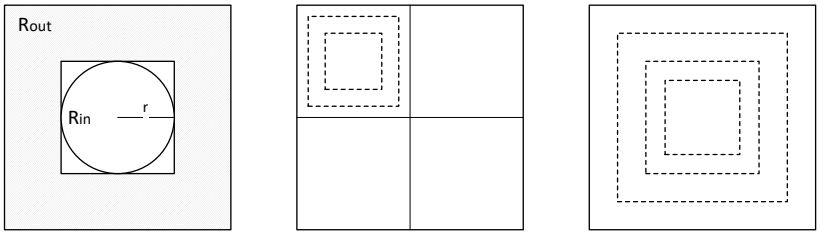
2 Related works

As stated ball detection approaches in the context of humanoid soccer-playing robots are divided into two groups: single-stage detectors and multi-stage candidate-based classifiers.

Single-stage detectors. In [4] a CNN is presented that employs a full-sized $m \times n$ raw image to predict the center of the balls. It produces two m and n-dimensional vectors as output. These vectors are discrete distributions and indicate the probability that each row and column of the image hold the center coordinates of the ball. To let the network converge more quickly, they provided more solutions by using two normal distributions as the label. The major drawback of their network is the expensive inferencing that takes about one second to complete for each image. Moreover, there is no guarantee that the network produces two unimodal distributions. So the output must be post-processed to find the most promising row and column that show the

center location of the ball. A fully convolutional neural network is proposed in [5] that can localize n objects simultaneously in an image. It uses an encoder-decoder design to produce n heat maps. Each heat map is a two dimensinal probability distribution that should be maximized at the interested location of an object belonged to a specific class. The proposed network has a real-time performance on a powerful GPU. Also, the network does not predict the boundary of objects. Another similar approach for ball detection is presented in [6]. The output of the network is a heat map that shows high values for ball pixels while contains near zero values for non-ball pixels. It achieved near real-time performance on a modern CPU without processing other tasks of the robot. In [7] a more efficient encoder-decoder architecture is proposed that maps an input image into a full-resolution pixelwise classification. To decrease the computation load they have used depthwise separable convolutions and removed skip connections. Although, this network achieved some good results in near real-time performance on a low-power processor, it trained and evaluated on a limited data set that contains only one ball type. A real-time CNN-based object detection approach for resource-constrained robotics is presented in [8]. Before feeding the input image to the network, it transforms the image using the object geometry to form a Visual Mesh. This mesh has a constant sample density for the object in different distances and it significantly reduces the computational complexity. To train the network a semi-synthetic data set is used. The data set is generated using 360 degree high dynamic range images and physics-based rendering. The proposed approach reported consistent and accurate precision and recall over all data set. However, the generalization performance of the method in a real robot is not evaluated.

Candidate-based classifiers. In [9] candidates are white blobs that satisfy expected color histogram, shape, and size of the target ball. For each candidate, a Histogram of Gradients feature is calculated and then classified by a cascade AdaBoost [10] method. The training time of this classifier was about 10 hours that is an issue for on-site training during the competitions. Recently several neural network based classifiers are presented for classic ball detection in standard platform league [11-13]. In [11] ball candidates are examined using black pentagons of the ball. The topology and configurations of the proposed network are optimized using a genetic algorithm. A more general candidate generation is suggested by [12]. For each interested pixel, a Difference of Gaussian filter with a kernel size that is proportional to the expected ball radius at the location of the pixel is applied. Then highly responded blobs are fed to a cascade of two CNNs that the first one reduces the number of proposals to five and the second one performs accurate classification. In [14] authors presented a simple and fast method to find the candidates. They divide the input image into a grid and then if the number of white pixels in each cell exceeds a threshold, it is considered as a ball candidate. They have investigated two CNNs for classification of the candidates. To accelerate the learning process they employed pre-trained networks and only retrained last layers for ball classification. Using a fixed grid for the whole image can lead to a candidate that either covers a portion of the ball or contains a small ball.



(a) Ball template (b) 2×2 feature map (c) 1×1 feature map

Fig. 1. (a) A template used to calculate importance weight of a particle in the ball proposed region detection algorithm. Inside region of the ball is approximated by the white box and outside region of the ball is illustrated by gray region. (b) and (c) show two feature maps and anchor boxes (dashed boxes).

3 Ball regions proposal

The main goal of our region proposal algorithm is to find some bounding boxes, most likely containing a full ball in the image window. Thanks to the accurate localization of our network introduced in the next section, we have no concern in fitting a region to the boundary of the ball. Our method is partially influenced by a simple ball detection method introduced in [15]. In contrast, we have proposed an iterative method that quickly converges to some region of interests using an easy to implement and efficient to compute feature. This method requires to analyse the white and green mask of the image. In this work, we have used a learned lookup table to extract these two masks. However, to create a white mask we can use a thresholding method on the brightness channel. Also, the green mask could be generated by activating pixels fall into a predefined range on the hue channel [16].

Our approach is presented in Algorithm 1. In line 1 we initialize n particles randomly at locations labeled as white. To accelerate converging and reduce the false regions, we have considered only white pixels located in a given distance (approximately eight meters). Each particle $\boldsymbol{x} = [cx, cy, r, w]$ is a vector representing the parameters of a circle. The scalar r is the roughly estimated radius of a ball located at coordinates (cx, cy) of the image. It can be derived using the camera matrix. An importance weight w is assigned to each particle that shows the likelihood of a ball. To estimate this weight we used a simple feature that can be computed efficiently. Considering a soccer ball on the playing field, we expect significant white pixels inside the ball (approximated by R_{in} in **Fig. 1** a). On the other hand, the region outside of the ball (approximated by R_{out} in **Fig. 1** a) should contains non-white pixels, especially green pixels. Following this feature analysis, we compute the weight of each particle in line 4 to 8. The function $n_c(R)$ calculates the number of pixels having color c inside the region R. This can be calculated efficiently considering a rectangular area around R using the integral image [17, 18]. Also, $1(expr)$ returns 1 if $expr$ interpreted as $true$ otherwise returns 0. The particles are converged through line 2 to 11. Each particle is replicated with a probability proportional to its weight in a resampling

Algorithm 1: Ball Regions Detection

1: X = Draw $\{x_i\}_{i=1}^{n}$ from white mask;
2: **for** $itt = 1$ *to* m **do**
3: **for** $i = 1$ *to* n **do**
4: R_{in} = region inside x_i;
5: R_{out} = region outside of x_i;
6: $w_{in} = \frac{n_{white}(R_{in})}{area(R_{in})}$;
7: $w_{out} = \frac{n_{not\_white}(R_{out})}{area(R_{out})} \times 1(\frac{n_{green}(R_{out})}{area(R_{out})} > green_{threshold})$;
8: $x_{i,w} = w_{in} \times w_{out}$;
9: **end for**
10: X = Resample $\{x_i\}_{i=1}^{n}$ with probability $\{x_{i,w}\}_{i=1}^{n}$;
11: **end for**
12: ValidCandidates = $\{x_i \mid x_{i,w} > w_{threshold}\}_{i=1}^{n}$;
13: ClusteredRegions = cluster(ValidCandidates);
14: **return** top_k(ClusteredRegions);

phase (line 10). In this phase, also every replicated particle is randomly translated and scaled to explore the neighborhood. Moreover, this random transformation makes finding candidates less sensitive to an accurate camera matrix. We repeat the weighting and resampling process until particles converge to some regions that may contain a ball. To reduce the number of particles we only keep the candidates with an importance weight exceeding a threshold (line 12). Then particles are clustered using an approach similar to non-maximum suppression. In contrast, we take an average of non-maximums to generate a cluster instead of suppressing them. This clustering method has a time complexity of $O(n^2)$. When the number of particles is large, a clustering algorithm with linear time complexity can be used in our method [19]. Since we have used a simple feature, it is likely that in some cases we miss the whole ball region; therefore, during the clustering phase, we increase the scale of the bounding box of each candidate, by a factor of two. To limit the maximum number of final regions we select k clusters that maximize the average importance weights (line 14).

4 Network Design and Training Phase

4.1 Design Phase

Here we propose a deep CNN architecture for ball detection, applicable in the region of interests, obtained in the previous step. To design the network we considered two main goals. Firstly, the network has to be fast in an embedded system with low computational resources. Secondly, it must be able to detect balls with different scales at a high precision rate. To achieve this network we inspired by the single shot multi-box detector (SSD) [2] and MobileNets [20] networks. SSD is an object detection network that uses some auxiliary multi-scales feature maps to detect objects at different sizes.

Each feature map is tiled with some predefined anchor boxes in a convolutional way and produces a fixed set of boxes with per-class scores for each box. MobileNets is an efficient model for embedded systems which is established on separable convolutional blocks. A separable convolutional block is a depthwise convolution followed by a 1×1 convolution. A 3×3 separable convolution is 8 to 9 times faster than a normal convolution without significant reduction in its accuracy [20].

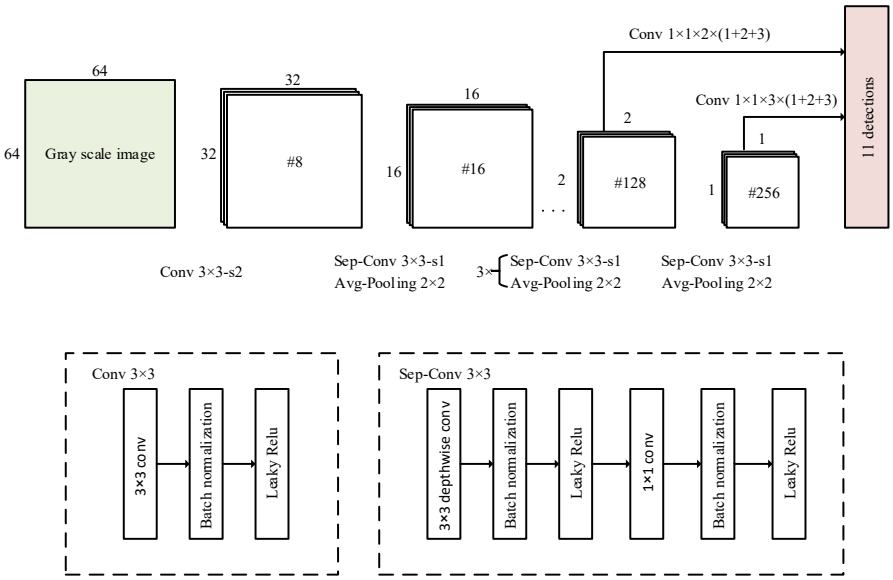


Fig. 2. The architecture of our deep convolutional network (BallNet) for ball detection (top). The microarchitecture of 3×3 normal and separable convolutions are shown in the bottom.

Our network is illustrated in **Fig. 2**. It merely consists of some low cost convolutional and average pooling layers. To compress information of the input image efficiently we downsampled the activation maps by a factor of two while doubling the number of channels. After all convolutions, we applied batch normalization (except for the convolutions applied to the feature maps). By normalizing the inputs of a layer, batch normalization acts as a regularizer and often eliminates the requirement of dropout [21]. In our case, this speeds up the training process while yielding consistent accuracy. To detect the balls, we have employed two last activation maps of the network as feature maps. At every $m \times n$ feature map with k associated anchor boxes to each cell, we produce $k \times m \times n$ detections. In this work, all anchor boxes are square shaped and located at the center of the feature map cells. Each detection can be considered as a vector of seven scalars, in which each element is predicted by applying a 1×1 convolutional filter. To express the bounding box of a detection d_i we use a tuple $l_i = \left(l_i^{cx}, l_i^{cy}, l_i^{w}\right)$, where l_i^{cx} and l_i^{cy} are scale-invariant translations of the center coordinates and l_i^{w} is the log-space translation of the width relative to the corresponding anchor box a_i. Also, a tuple of two scalars $p_i = (p_i^{ball}, p_i^{bg})$ is used to indicate the

scores of the ball and background classes. The scalar c_i shows the confidence score of a detection. It reflects the confident level that the predicted bounding box fits the boundary of a ball. There are two anchor boxes related to each cell of the first 2×2 feature map with scales of 0.25 and 0.4 (**Fig. 2** b). The second feature map uses three anchor boxes with scales of 0.33 , 0.5 and 0.75 (**Fig. 2** c). These scales and number of anchor boxes are determined so that the first feature map is more responsible for balls located at the corners, and the second one is more responsible for center located balls.

4.2 Training Phase

Training of the introduced network involves matching strategy of anchor boxes, optimization of a multi-task cost function, and preparing the dataset.

Assignment of anchor boxes. To match anchor boxes with an annotated label we follow SSD [2]. With each sample containing a ball, we first calculate the Jaccard overlap (also known as intersection over union) between the annotated box and each of the anchor boxes. Note that the location and size of the anchor boxes in the feature maps and the annotated box in the sample image are normalized. Then the anchor box with the best score in Jaccard overlap is selected as the best matching anchor box. As described in [2] we also match anchor boxes with Jaccard overlap above a value 0.5 to ease the learning procedure.

Loss function. The loss function in our algorithm is based on both SSD [2] and YOLO [3]. We define the loss function as an ensemble of three terms named localization loss (L_{loc}), classification loss (L_{cls}), and confidence loss (L_{conf}):

$$L_{total} = \frac{1}{N}(\alpha L_{loc} + \beta L_{cls} + \gamma L_{conf}) \tag{1}$$

In which $\alpha, \beta$and γ specify the contribution of the loss of each task and is determined by cross validation. N is the number of anchor boxes matched with the ground truth box. The localization loss counts only for the matched anchor boxes with the ground truth box:

$$L_{loc}(l, g) = \sum_{i \in \text{pos}}^{N} \sum_{m \in \{cx, cy, w\}} |l_i^m - \hat{g}_i^m| \tag{2}$$

$$\hat{g}_i^{cx} = \frac{(g^{cx} - a_i^{cx})}{a_i^w} \qquad \hat{g}_i^{cy} = \frac{(g_i^{cy} - a_i^{cy})}{a_i^w} \tag{3}$$

$$\hat{g}_i^w = \log(\frac{g_i^w}{a_i^w}) \tag{4}$$

where l_i is the predicted box related to the selected anchor box a_i and $\hat{g}_i$ is the ground truth box g encoded with respect to a_i. The classification loss is the sum of ball class losses for selected anchor boxes (i.e. positive boxes) and background class losses for anchor boxes which are not matched (i.e. negative boxes):

$$L_{cls}(p) = -\sum_{i\in\text{pos}}^{N} log(\hat{p}_i^{ball}) - \sum_{i\in\text{neg}}^{M} log(\hat{p}_i^{bg}) \quad (5)$$

where M is the number of negative boxes used. To balance between positive and negative examples we set M as twice of N at most. The negative boxes with high predicted scores for ball class are selected first. Also $\hat{p}_i^c$ is the softmax of a class $c \in \{ball, bg\}$:

$$\hat{p}_i^c = \frac{exp(c_i^c)}{\sum_c exp(p_i^c)} \quad (6)$$

We expect the confidence c_i of a positive box conveys the Intersection Over Union (IOU) between the decoded predicted box $d(l_i)$ and the ground truth g. So we defined the confidence loss as follows:

$$L_{conf}(c) = \sum_{i\in\text{pos}}^{N} \left| c_i - IOU_{d(l_i)}^{g} \right| \quad (7)$$

Optimization. To optimize the loss function we have used Adam optimizer with a piecewise constant schedule for learning rate. In our experiments, we begin with 1e-3 as starting learning rate. We perform a training phase for 5k iterations followed by reducing the learning rate to 1e-4 and then continuing the training for 10k iterations. We lowered the learning rate once more to 1e-5 for 3k more iterations. We also tried other learning rate schedules like poly and exponential learning rate decay methods but we have found the piecewise constant schedule works better for our application.

We also set α, β and γ to 5, 0.6 and 1 respectively. In our experiments, we found the classification task is relatively easier than localization to learn. To avoid overfitting in the classification task we set β to 0.2 after 7k iterations of training.

Data set. One of the most important aspects of modern object detection systems is the data set. To prepare the data set we annotated 1k images from various RoboCup competitions and our research lab, including 5 different types of balls and variety of negative candidates like robots, goal posts, penalty marker, humans, and other objects around the field. In the data set, all balls are located approximately at a distance up to 4 meters away from the camera. Then we generated 64×64 patches by applying the random crop, random scale, mirroring and random brightness around ball regions and other regions of the annotated images. To augment negative patches with more diverse regions we manually created some patches around goal posts, lines, and penalty markers. Our training set after augmentation contains 12k images. Some positive and negative samples are shown in **Fig. 3** rows 1-4.

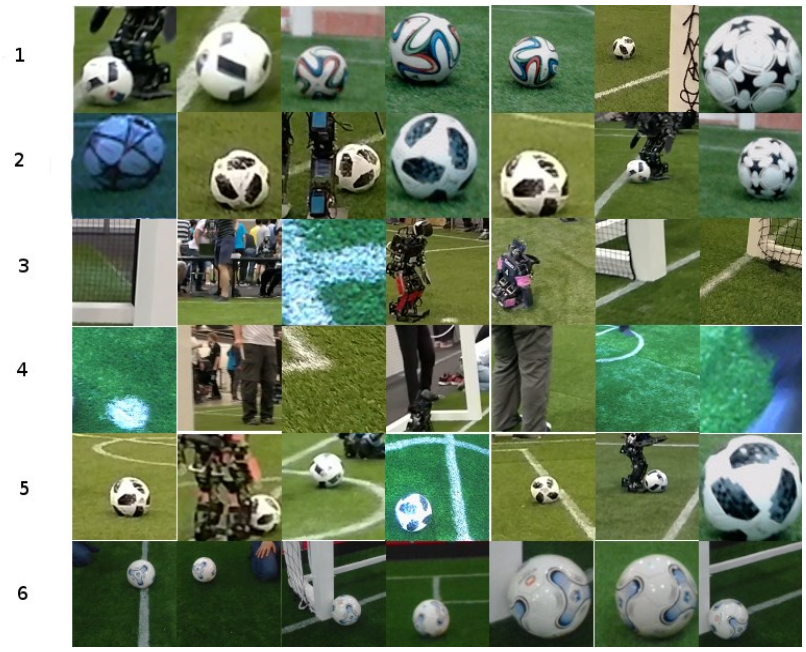


Fig. 3. Row 1, 2: Positive samples in the training data set. Row 3 and 4: Negative samples in the training data set, Row 5: Positive samples of the validation data set. Row 6: Positive samples of generalization data set with a new ball pattern not used for training.

4.3 Inference

At the inference stage, we assume that the category of a detection is the class with the maximum predicted score. Therefore it is possible that several detections report the ball class. Among these detections, we select a detection d_i maximizing the localization score $s_i = p_i^{ball} \times c_i$. As mentioned before the network only predicts a set of offsets to the assigned anchor box, hence to get final bounding box we decode the predicted bounding box l_i:

$$d^{cx}(l_i) = (l_i^{cx} * a_i^w) + a_i^{cx} \quad d^{cy}(l_i) = \left(l_i^{cy} * a_i^w\right) + a_i^{cy} \tag{8}$$

$$d^w(l_i) = d^h(l_i) = exp(a_i^w) * a_i^w \tag{9}$$

5 Experiments

In this section, we evaluate the performance of the ball regions detection algorithm. Then we show the effectiveness of the designed neural network under different conditions. Moreover, the overall performance of the proposed ball detection method is investigated in a real-world test. All experiments are performed in a kid-size humanoid robot [22] equipped with a Logitech c920 camera capturing 640×480 images at a

rate of 30 frames per second and an Intel NUC with a Core i3 U6100 processor. The training is carried out on a Nvidia GTX 1080 GPU.

5.1 Region proposal analysis

We evaluated our *region proposal algorithm* on 400 images acquired from various conditions including moving, occluded, on the line, and near the goal post balls while the robot is either walking or standing. Then we initialized the algorithm with 500 particles and reduced it to 300 particles in the resampling step. After only two iterations particles are converged. We noticed that in 96% of images our algorithm finds at least one region that contains a full ball and in 4% of images, the clustered regions cover a portion of the ball. The step by step process of the algorithm is shown in **Fig. 4**. As shown in column three, our algorithm can detect regions with different ball patterns and sizes without any modification. **Table 1** represents the run-time cost of the algorithm. The algorithm takes 1.46 milliseconds in average that indicates it as a fast and real-time region proposal method.

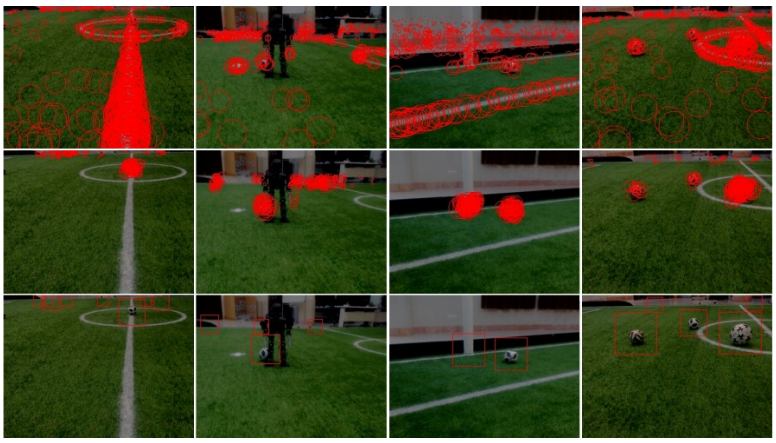


Fig. 4. The process of generating the ball regions. Top row: initialized particles. Middle row: converged particles after two iterations. Bottom row: Final clustered regions.

Table 1. Run-time cost of ball region proposal algorithm in milliseconds.

Steps	Max	Average
Creating integral images	1.48	0.5
Converging particles	4.19	0.88
Clustering	1.22	0.08

5.2 Model analysis

To evaluate our network (the BallNet) we prepared two datasets, the first set contains 2k images of both positive and negative samples with the same ball patterns existed in training dataset (**Fig. 3** row 5); The second set contains 1k of only positive samples

with a new ball pattern that are never used in training (**Fig. 3** row 6). The second set tests the generalizability of our model. This is important because, in the humanoid soccer league, participating teams may face with new ball patterns in every RoboCup competition. Also, to investigate a speed-accuracy trade-off we trained a lighter network (BallNet-mini) in which all activation maps are halved compared to the BallNet.

Validation. Similar to other researchers we use the most common metrics of accuracy, recall rate, and precision rate to evaluate our classifiers. However, in the RoboCup soccer leagues, misclassification of the ball can lead to a score loss. Therefore we also analysed false positive rate. As can be seen in **Table 2** the proposed BallNet model benefits from a high precision and recall rate while maintaining a low false positive rate, which means the trained model is able to accurately detect balls in most regions covering a ball. As mentioned earlier, the classification task is rather easier than the localization. Therefore the BallNet-mini model reported promising metrics, although all the metrics degraded slightly.

Table 2. Classification performance of the proposed networks

Model name	Accuracy	Precision rate	Recall rate	FP rate.
BallNet	96.43	97.17	94.29	2.00
BallNet-mini	94.84	95.71	91.89	3.00

To evaluate the performance of the networks in the localization task we used average IOU and precision with IOU levels more than 0.5, 0.75 and 0.9. Note that only the samples labeled and classified as ball are used to calculate the metrics. **Table 3** shows the high accuracy of BallNet model localization. As expected the BallNet model with more capacity outperforms the smaller model in localization task. **Fig. 5** shows the results with different IOU levels.

Table 3. Localization performance of the proposed networks

Model Name	Avg. IOU	P:0.5	P:0.75	P:0.9
BallNet	0.788	96.23	66.70	9.17
BallNet-mini	0.763	92.86	56.95	7.84

Generalization. Since there are no negative samples in the generalization set, we only used detection rate (number of samples detected as positive divided by the number of positive samples in the data set) to evaluate the classification performance of our models. As summarized in **Table 4** the BallNet significantly outperforms the BallNet-mini at the generalization of both classification and localization tasks. Although the detection rate is significantly decreased in the BallNet, the result shows that there is a high possibility of increasing the model performance with fine-tuning during the competitions.

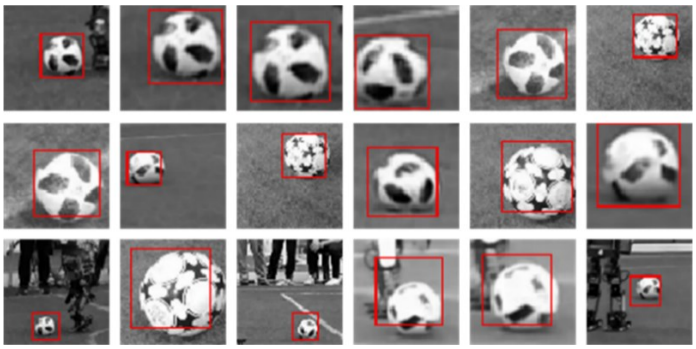


Fig. 5. The result of the ball detections with different IOU levels of P:0.9, P:0.75, and P:0.5 illustrated in the top, middle, and bottom row, respectively.

Table 4. The performance of the models on the generalization set.

Model Name	Detection rate	Avg. IOU	P:0.5	P:0.75	P:0.9
BallNet	70.9	0.768	94.30	69.2	10.11
BallNet-mini	53.53	0.69	90.0	40.75	3.20

Time complexity. We have evaluated the speed of the models at inferencing and training phases. As shown in **Table 5**, both models are very fast. Although we have halved the BallNet-mini, the inference time has not linearly decreased compared to the BallNet. The discrepancy may be more highlighted in a weaker computing device.

Table 5. Time and parameters complexity of the models.

Model Name	Inference (ms)	Train (minute)	Parameters	GFLOPS
BallNet	1.782	17.04	65,166	133,319
BallNet-mini	1.198	16.30	21,686	44,855

5.3 Overall performance

To evaluate the overall performance of the presented pipeline, we saved the input images, detected regions, and predictions of the BallNet once in every 10 frames from a 5-minute real gameplay. The resulting measurements of the classification were as follows: 91.43% accuracy, 89.72% precision rate, 80.07% recall rate, and 3.30% of false ratio. Also, we measured a runtime cost of 5.13 milliseconds in average for the entire detection pipeline while other tasks of the robot were running in parallel. As can be seen, due to different lighting conditions and more blurred images caused by robot movements, we experienced a weaker performance.

Conclusion

In this paper, we presented a fast and accurate ball detection pipeline for humanoid soccer playing robots. The first stage of the proposed method reliably extracts some regions that may contain a full ball and then the designed deep neural network (called the BallNet) predicts the exact location of the ball in each region, with a high recall and precision rate. We demonstrated that our region proposal algorithm can deal with different ball types without further modifications. The proposed pipeline also fulfilled the computational constraints of low-cost humanoid robots. The method is fast to train (about 17 minutes) and in average takes less than 5 milliseconds to run on a standard Core i3 CPU. The pipeline can be applied for detecting other objects with known sizes like the goal posts and other robots. Our data sets and TensorFlow implementation of the network is publically available online[1] for other researchers and future works.

References

1. Zhao, Z., Zheng, P., Xu, S., Wu, X.: Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems 1-21 (2019)
2. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: Single Shot MultiBox Detector. In: Computer Vision – ECCV 2016, pp. 21-37. Springer International Publishing, (2016)
3. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788. (2016)
4. Speck, D., Barros, P., Weber, C., Wermter, S.: Ball Localization for Robocup Soccer Using Convolutional Neural Networks. In: RoboCup 2016: Robot World Cup XX, pp. 19-30. Springer International Publishing, (2017)
5. Schnekenburger, F., Scharffenberg, M., Wülker, M., Hochberg, U., Dorer, K.: Detection and Localization of Features on a Soccer Field with Feedforward Fully Convolutional Neural Networks (FCNN) for the Adult-Size Humanoid Robot Sweaty. In: Proceedings of the 12th Workshop on Humanoid Soccer Robots, 17th IEEE-RAS International Conference on Humanoid Robots, pp. 1-6. IEEE, (2017)
6. Speck, D., Bestmann, M., Barros, P.: Towards real-time ball localization using cnns. In: Proceedings of 22nd RoboCup International Symposium. (2018)
7. van Dijk, S.G., Scheunemann, M.M.: Deep learning for semantic segmentation on minimal hardware. arXiv preprint arXiv:1807.05597 (2018)
8. Houliston, T., Chalup, S.K.: Visual Mesh: Real-time Object Detection Using Constant Sample Density. arXiv preprint arXiv:1807.08405 (2018)
9. Farazi, H., Allgeuer, P., Behnke, S.: A monocular vision system for playing soccer in low color information environments. arXiv preprint arXiv:1809.11078 (2018)

[1] https://github.com/mrl-hsl/cnnBallDetector

10. Rezaei, M., Klette, R.: Computer Vision for Driver Assistance: Simultaneous Traffic and Driver Monitoring. Springer International Publishing, Cham (2017)
11. Felbinger, G.C., Göttsch, P., Loth, P., Peters, L., Wege, F.: Designing Convolutional Neural Networks Using a Genetic Approach for Ball Detection. In: Proceedings of 22nd RoboCup International Symposium. (2018)
12. Leiva, F., Cruz, N., Bugueño, I., Ruiz-del-Solar, J.: Playing Soccer without Colors in the SPL: A Convolutional Neural Network Approach. arXiv preprint arXiv:1811.12493 (2018)
13. Menashe, J., Kelle, J., Genter, K., Hanna, J., Liebman, E., Narvekar, S., Zhang, R., Stone, P.: Fast and Precise Black and White Ball Detection for RoboCup Soccer. In: RoboCup 2017: Robot World Cup XXI, pp. 45-58. Springer International Publishing, (2018)
14. Gabel, A., Heuer, T., Schiering, I., Gerndt, R.: Jetson, where is the Ball? - Using Neural Networks for Ball Detection at RoboCup 2017 (2018)
15. Hayashibara, Y., Minakata, H., Irie, K., Maekawa, D., Tsukioka, G., Suzuki, Y., Mashiko, T., Ito, Y., Yamamoto, R., Ando, M., Hirama, S., Suzuki, Y., Kasebayashi, C., Tanabe, A., Seki, Y., Masuda, M., Hirata, Y., Kanno, Y., Suzuki, T., Supratman, J., Machi, K., Miki, S., Nishizaki, Y., Kanemasu, K., Sakamoto, H.: CIT Brains KidSize Robot: RoboCup 2015 KidSize League Winner. In: RoboCup 2015: Robot World Cup XIX, pp. 153-164. Springer International Publishing, (2015)
16. Mühlenbrock, A., Laue, T.: Vision-Based Orientation Detection of Humanoid Soccer Robots. In: RoboCup 2017: Robot World Cup XXI, pp. 204-215. Springer International Publishing, (2018)
17. Teimouri, M., Fatehi, A., Mahmudi, H., Sagharichi Ha, P., Gholami, A., Delavaran, M.H., Movafegh, F., Rahmani, G., Fathi, E.: MRL Team Description Paper for Humanoid KidSize League of RoboCup 2018 (2018)
18. Rezaei, M.: Computer Vision for Road Safety: A System for Simultaneous Monitoring of Driver Behaviour and Road Hazards. (2016)
19. Laue, T., Röfer, T.: Pose Extraction from Sample Sets in Robot Self-Localization-A Comparison and a Novel Approach. In: Ecmr, pp. 283-288. (2009)
20. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
21. Normalization, B.: Accelerating deep network training by reducing internal covariate shift. CoRR.–2015.–Vol. abs/1502.03167.–URL: http://arxiv. org/abs/1502.03167 (2015)
22. Mahmudi, H., Fatehi, A., Gholami, A., Delavaran, M.H., Khatibi, S., Alaee, B., Tafazol, S., Abbasi, M., Yeghane Doust, M., Jafari, A., Teimouri, M.: MRL Team Description Paper for Humanoid KidSize League of RoboCup 2019 (2019)