# Jetson, Where Is the Ball? Using Neural Networks for Ball Detection at RoboCup 2017

Alexander Gabel(✉), Tanja Heuer, Ina Schiering, and Reinhard Gerndt

WF Wolves, Ostfalia University of Applied Sciences, Wolfenbüttel, Germany
{ale.gabel,ta.heuer,i.schiering,r.gerndt}@ostfalia.de

**Abstract.** The approach of using neural networks in the RoboCup humanoid league for ball detection is investigated in a case study at the RoboCup 2017 competition. A patch-based classification approach is used. Two different ConvNet architectures, the Inception v3 network by Google and AlexNet are evaluated in the context of a ROS-based architecture on a robot with a Jetson GPU board. The aim is to allow for an efficient re-training of neural networks in the context of the competition.

**Keywords:** Ball detection · Neural network · Humanoid league · RoboCup

## 1 Introduction

Based on the intention of RoboCup Humanoid league, that in 2050 "a team of autonomous humanoid robots shall play soccer against the human world champion" [5], the complexity of the tasks of RoboCup are increasing. Examples of this increasing complexity are the use of a standard FIFA ball, increasing field sizes and the use of artificial grass. These challenges lead to increasing requirements concerning object detection.

The WF Wolves team previously used the Haar cascade algorithm [21]. This algorithm is integrated in frameworks as OpenCV and allows for a very time-efficient ball recognition when the ball is in medium range. It is not able to cope with far away balls or partial occlusion. In such situations, which will occur more often because of the growing field size and an increasing number of players, the prior approach is unable to reliably detect balls in larger distances (Fig. 1).

A promising approach is the use of neural networks, which were recently investigated in the context of the humanoid league [18]. However, in the past, neural networks, particularly fully-connected ones, were too computationally intensive for usage in image classification. Due to improved architectures (i.e. convolutional neural networks - CNN) and adapted hardware, this has changed in the last decade. However, the usage of deep neural networks for end-to-end learning in the context of the humanoid league was not feasible until the recent introduction of fast embedded GPUs, for (small) mobile robot platforms.


D. Holz et al. (Eds.): RoboCup 2018, LNAI 11374, pp. 181–192, 2019.
https://doi.org/10.1007/978-3-030-27544-0_15

**Fig. 1.** Far balls detected by CNN (pink circles), false positive by Haar Cascade classifier (violet circle). Red line: approximated field outline. Blue squares: patches analysed by CNN. Blue rectangles: goal post hypotheses (Color figure online)

Our approach is based on pre-trained convolutional neural network models, which are fine-tuned[1] to the RoboCup ball classification scenario. Advantages of this approach include vastly reduced training times, higher accuracy and less overfitting, when compared to a model trained from scratch. To localize the ball, not the whole image is classified by the CNN, but the image is divided into patches, which are filtered based on color information.

In this paper the use of neural networks for ball detection, in particular far away balls, is evaluated based on experiences of the RoboCup 2017 competition. The aim of the proposed approach is a CNN based classifier, which is fast-adaptable to on-site conditions. We present the adapted vision pipeline, the machine learning approach and experimental results.

## 2 Related Work

Neural networks are a widespread used technology for object recognition. In the RoboCup soccer league neural network based systems for object recognition have mainly been used in the middle size league. Mayer et al. [12] proposed a neural network for robot detection on the field. Their approach uses the classical feature extraction/classification separation with hand-crafted features designed by humans. Our approach rather uses the more recent, but also more computationally intensive approach of end-to-end learning (deep learning), which also learns the feature extraction from the raw pixels. Furthermore neural networks are sometimes used in combination with other approaches, such as Kalman filters [11,20] to improve performance for objects in motion (i.e. predict their position).

[1] https://cs231n.github.io/transfer-learning/.

At present, neural networks become computationally feasible also for the humanoid league and standard platform league for ball and opponent recognition. In the humanoid league the team Hamburg Bit-Bots [18] proposed a CNN approach for ball localization. They used a custom network architecture trained from scratch by using a normal distribution as teaching signal for each coordinate axis. The team AUTMan [6] presented the idea of a CNN model for recognizing opponents. In the Standard Platform League the approach using neural networks was also recently investigated by several teams [1,3,14,15].

CITBrains from Japan are the only team stating the use of CNNs for ball, goal post and opponent recognition for the RoboCup competition in their team description paper for the RoboCup 2017 [16]. Hence until now mainly preliminary lab results about the use of CNNs in the humanoid league are described. Besides the RoboCup, CNNs implemented on a Jetson TX1 board is currently a gladly used approach in a low-power environment [2,13]. The CNN AlexNet is retrained for car plate and person recognition with satisfying results [4,10]. In the following we present our results and experiences from using CNNs during the RoboCup 2017.

## 3 Vision Pipeline

In general, we use a patch-based classification approach, to decide whether a particular region of interest (ROI) in the image contains a (partial) ball or not. To increase computational efficiency, classification is only applied to patches, which potentially contain a ball. Therefore the image is preprocessed and filtered (Fig. 2).

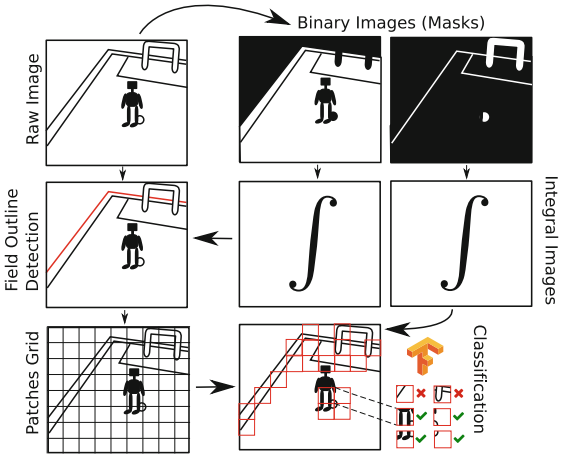


**Fig. 2.** Vision pipeline (Color figure online)

First we generate binary images (masks) by classifying the color of each pixel in the source image. For ball detection the masks for soccer *field green* and for *ball white* are relevant.

To keep rectangular sum calculations in the mask efficient, we transform the masks into integral images. Based on the detection of field outlines, regions outside of the field, without touching it, are not considered as a possible ball region. The image is then divided into a grid of $n \times n$ patches with a default patch size $n$ of 50 pixels (Fig. 5). This hyper-parameter was chosen as a performance/accuracy-trade-off.

For each patch, we determine the number of pixels belonging to the *ball white* color class using the integral image. Patches where this number exceeds a predefined threshold (default: 30), are considered as a *region of interest* (ROI) for classification.

Each of those ROIs is then fed into the classifier, a deep convolutional neural network, which calculates probabilities for the two classes *ball/partialball* and *noball*. The classifier itself is running as a separate ROS[2] node.

## 4 Machine Learning

We evaluated two different ConvNet architectures, the modern Inception v3 network by Google [19] and AlexNet [9], one of the first well-performing deep convolutional neural networks. As we use TensorFlow for our model, all operations in the training refer to the corresponding functions and implementations in TensorFlow (Fig. 3).

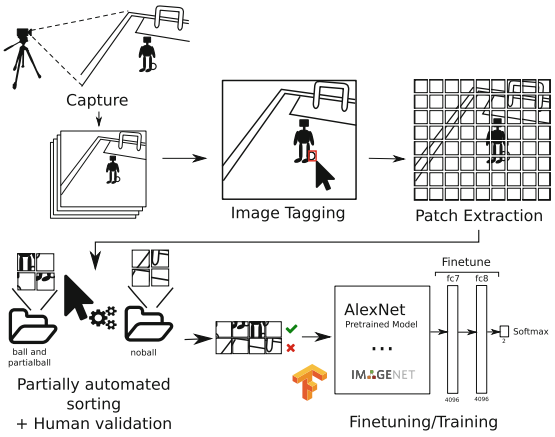


**Fig. 3.** Machine learning pipeline

For both evaluated networks, we did not perform a full training of the network, as the goal during the competition was to be able to relatively fast adjust

[2] Robot Operating System (http://www.ros.org).

the network to on-site conditions. Hence we fine-tune models of each network, which were pre-trained for ImageNet. Also these models should be able to generalize better, as they already learned common features for object recognition and overfitting is less likely to occur, as we only retrain the last layer(s). For fine-tuning Inception v3, there is already a script and tutorial provided by TensorFlow[3]. Also for AlexNet there exists a project for fine-tuning[4], where the pretrained model is converted from the Caffe [7] model of BVLC AlexNet[5].

For collecting training images, we use a regular tripod in the height of our robot with the camera of our robot. We capture single images instead of the whole data stream to include common situations, which might also occur during a real game, while avoiding too many pictures of a single perspective, which would introduce an unwanted bias to our dataset, and look for difficult situations like other white objects on/near the field.

After capturing, the images are labeled using the ImageTagger[6] by the Hamburg Bit-Bots team, which allows to build up a shared image database for humanoid soccer competitions. In the tagger application, we annotate balls in the image using bounding boxes.

The images are then divided into small patches (see Fig. 5). To generate training data, no patches are filtered out in this step, as this has proven to result in a better, more flexible model. In an automated labeling patches are labeled as *ball* if the overlap of any ball bounding box with the region is larger than a certain threshold.

As this automated sorting is still error-prone, after this step a human has to validate the resulting patches and potentially to move images to the correct folder. Files where even the human is unsure about classification are removed from the dataset.

In general we do not perform additional preprocessing of the training/validation images (i.e. translation/noise addition/etc.) to reduce the time needed for on-site training. Due to the high amount of patches, we expect to have enough translation variety already present in the dataset.

As the training data is highly unbalanced between the classes, i.e. we have a lot of negative samples, but comparably only a very small percentage of positive (ball) samples, the training process has to be adjusted. Otherwise the classifier might prefer the overrepresented class over the other. This effect is called accuracy paradox.

There are multiple ways of handling imbalanced classes, including oversampling, undersampling, class weights or balanced batches. To cope with the highly imbalanced training data, instead of using *oversampling* on the *ball* class *undersampling* on the *noball* class was investigated. This led to an increased false positive rate due to less negative (*noball*) samples. Using alternatively class weights

[3] https://www.tensorflow.org/tutorials/image_retraining.
[4] https://kratzert.github.io/2017/02/24/finetuning-alexnet-with-tensorflow.html.
[5] https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet.
[6] https://github.com/bit-bots/imagetagger.

led to a more robust model with a reduced false positive rate. Also balanced batches yielded good results for the Inception network.

For fine-tuning AlexNet, we re-train the last two fully-connected layers (fc7 and fc8). Softmax cross entropy is used as loss function. Batches of 128 randomly chosen patches are fed into the network for training. Adam [8] is used as optimization procedure and TensorBoard for monitoring the training and validation accuracy, as well as the cross-entropy loss over time.

## 5 Experimental Results

The presented approach is evaluated in the following based on the experience of the RoboCup 2017. A description of the dataset used for training is provided and the performance of the inference process is investigated based on the built-in Jetson board of the robot.

The proposed CNN-based method (pink circles) is able to detect partially covered balls and far away balls, as shown in Fig. 1, with less false-negatives compared to the Haar cascade approach (violet circles) (Fig. 4).

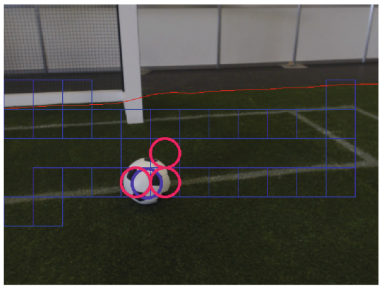

**Fig. 4.** Close ball detected by Haar Cascade classifier and CNN (Color figure online)

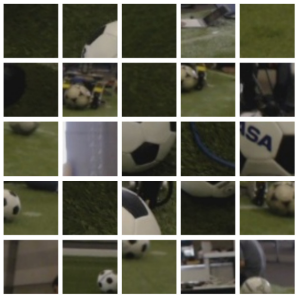

**Fig. 5.** Examples of patches (non-filtered)

### 5.1 Dataset

The images used as the basis for our dataset were recorded at 3 different locations: our lab test field, German Open 2017 and RoboCup 2017 Japan. The original ∼1800 images had a resolution of $640 \times 480$ respectively $640 \times 360$ pixels and were all taken with our robot's camera.

The dataset consists of ∼158k patches, where ∼3k were labelled as *ball or partial ball*, and the rest ∼155k patches were labelled as *no ball*. The images contain different kinds of white balls, with difficult situations, such as partial occlusion by robots, sun light from open doors and various shaped and sized objects of white color. We particularly included problematic situations on the RoboCup, where we experienced a lot of false positives.

For evaluation of the different models, we captured an additional dataset consisting of ∼1000 images. This dataset, in this paper referenced as 'evaluation dataset', is used for performance testing on whole-frame pictures, as well as for the confusion matrix.

### 5.2 Validation

We evaluated several models varying training epochs/steps[7], validation split and class imbalance handling (Table 1). The true positive rate is defined as the number of correctly detected balls divided by the number of ground truth balls. The false discovery rate is defined as the number of false positive balls divided by the sum of false positive and true positive balls.

**Table 1.** Evaluated models with true positive rate (TPR), false discovery rate (FDR), validation accuracy and training duration (inception has constant bottleneck creation time = only required once for all models). (AlexNet 20 epochs (10% validation data, class weights); Inception v3 500 steps (10% validation data, balanced batches); Inception 4000 steps (10% validation data, balanced batches); Inception 20000 steps (10% validation data, class weights); AlexNet 38 epochs (20% validation data, class weights); AlexNet 64 epochs (10% validation data, undersampling); AlexNet 9 epochs (10% validation data, class weights))

| Model | Haar | AlexNet 20 epochs | Inception 500 steps | Inception 4000 steps | Inception 20000 steps | AlexNet 38 epochs | AlexNet 64 epochs | AlexNet 9 epochs RC2017 |
|---|---|---|---|---|---|---|---|---|
| TPR | 24.135% | 74.382% | 96.34% | 96.736% | 98.811% | 72.908% | 88.131% | 68.546% |
| FDR | 7.925% | 10.9% | 10.724% | 14.06% | 80.697% | 18.568% | 39.346% | 12.278% |
| Val. acc. | - | 99.25% (N = 15776) | 98% (N = 100) | 88% (N = 100) | 93% (N = 100) | 99.48% (N = 15776) | 96.97% (N = 1076) | 99.14% (N = 15776) |
| Train. Dur. | - | ∼8, 5 h | 2, 5 h + 1 m | 2, 5 h + 7 m | 2, 5 h + 36 m | ∼16 h | ∼45 m | ∼4 h |

From the Table 1 it can be seen, that the Haar Cascade algorithm has a low TPR, therefore not detecting a large percentage of balls. AlexNet performs better, but with an increased FDR. We suspect that the rather high FDR for AlexNet 64 epochs, is due to less amount of training data caused by undersampling. Inception with 20000 epochs is probably overfitted. We have selected Inception 500 steps and AlexNet 20 epochs, as they have the best trade-off between true positives and false positives[8]. As can be seen in Fig. 6, the Haar cascade algorithm is unable to detect balls at distances larger than 3.50 m, in contrast to both CNN algorithms.

We achieved ∼99% validation accuracy during the RoboCup competition (AlexNet 9 epochs). During the RoboCup, the time available for training is very

[7] Number of batches (batch size 100) used for training.

[8] Additional results are left out because of page limitations, but can be downloaded from our website (https://www.wf-wolves.de/jetson-rc2017/).

constrained, therefore we only trained for 9 epochs, which already took about 4 h. The TensorFlow *retrain.py* script used for fine-tuning Inception, uses a technique to first generate *bottleneck files*, which are the results of processing each training image through the first layers (which are not changed). This leads to a constant start-up cost, and greatly reduces training time in total. The concept may be applied to AlexNet as well, which might decrease training time by a large portion. However, the false positive rate slightly increases for the CNNs.

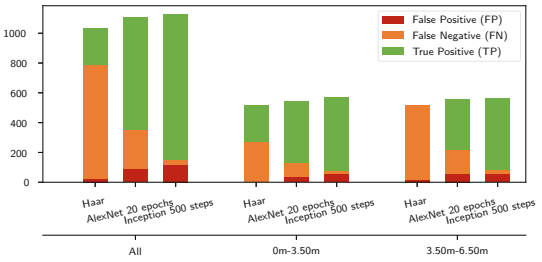


**Fig. 6.** Comparison of three different classifier models: Haar Cascade; AlexNet trained for 20 epochs with class weights; inception v3 trained for 500 steps with class weights

### 5.3 Performance of Inferencing

For the evaluation we investigated Inception v3 and AlexNet as described in Sect. 4. The Inception v3 model might in principle lead to better classification accuracy results, but was too slow for this use case. Figures 7 and 8 show the processing time required for the whole image pipeline (time between publishing of a camera image to the receiving of the debug image).

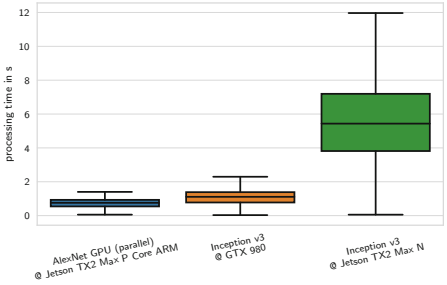


**Fig. 7.** Pipeline performance AlexNet vs Inception v3 (outliers excluded)

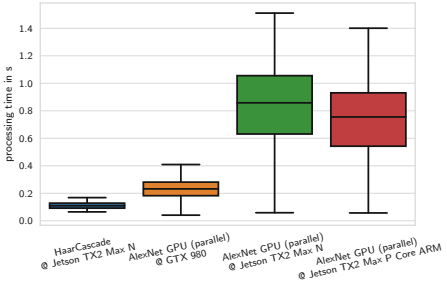


**Fig. 8.** Pipeline performance AlexNet vs Haar Cascade (outliers excluded)

Running the Inception network on a CPU, took about 1 s per inference run/patch. Running it on a Desktop GPU, increased the performance to about

0.2 s per patch, which is still slow, when having to handle a large number of patches. Therefore we switched to AlexNet, a more lightweight model, which still promised sufficient classification accuracies, while achieving better real-time performance. With AlexNet, we achieved approximately 0.1 s for the inference time per patch on a CPU. For larger amounts of patches this was still too slow. Using the Jetson TX2 GPU board the duration could be further reduced.

To further optimize the performance, we compared the sequential execution of the network with a parallelized version taking a variable batch of patches as input. The results for this can be seen in Figs. 9 and 10. While the processing time in the sequential version grows linearly, the parallel version has a relatively high amount of outliers. Based on mean execution times, the parallel version offers a huge advantage compared to the sequential method.

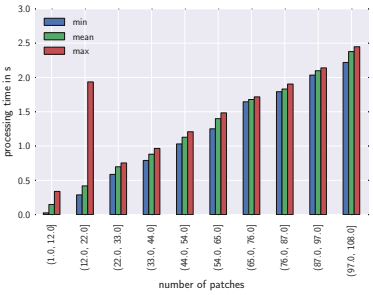


**Fig. 9.** Execution time of sequential AlexNet on the Jetson GPU

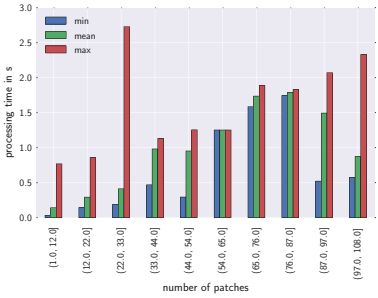


**Fig. 10.** Execution time of parallel AlexNet on the Jetson GPU

We used the *Max-P ARM* performance mode of the Jetson TX2, which was the most stable, but at the same time well-performing setting. We also investigated the *Max-N* mode, with which we had stability issues, which may have been caused by current limitations of the power supply. Table 2 gives an overview about the investigated performance modes.

**Table 2.** Tested performance modes of the Jetson TX2

| Mode | ARM | Denver | GPU | Power w/o GPU | Power w/ GPU |
|---|---|---|---|---|---|
| Max-P | 2.0 GHz | Disabled | 1.12 GHz | 0.6 A | 0.9 A |
| Max-N | 2.0 GHz | 2.0 GHz | 1.30 GHz | 1 A | >2 A |

## 6 Discussion

In general the presented approach was feasible during the competition and the accuracy and model performance is promising. An advantage of the patch-based

approach compared to approaches operating on the whole image is, that it is easier to generate an adequate amount of training data, as one image contains already many patches. Also training samples are usually better distributed, as e.g. the ball is not always in the center or often only partly visible. This improves especially the detection of partially covered balls. Still, there may be a bias from other sources, such as illumination and exposure. When choosing a patch size of $50 \times 50$ pixels with a stride-size of 50 pixels, we had an accuracy-performance trade-off in mind. Lowering the stride-size, will lead to more patches, which might simplify the problem for the classifier, but decreases the runtime-performance. Furthermore it might be worth to investigate adaptive patch sizes, for example based on the distances from the robot's view.

A potential disadvantage of the patch based approach are patches containing small parts of a ball without further context in the image, where even humans would not be sure. Whole-image models may need a longer training time, since there are typically no pre-trained models available, and the object localization problem is harder compared to a binary classification problem.

The general approach to fine-tune a pre-trained networks with a verified architecture offers several advantages. The risk of overfitting is reduced and training times are shorter, which is important for on-site training during competitions. However, publicly available pre-trained networks are often designed for a different, more generic use-case. In our case the pre-trained networks AlexNet and Inception v3 were designed for fixed input dimensions. Inception v3 uses $299 \times 299 \times 3$ and AlexNet $227 \times 227 \times 3$ as input dimensions. This is considerably larger than the patch size ($50 \times 50 \times 3$) chosen here. Therefore each patch has to be scaled up to the network's input dimension. This yields to a computational overhead, which would not be necessary in case of a custom designed network. Furthermore the parallelization on the Jetson may suffer from this, as there are more CUDA units necessary to compute the larger network.

To enhance the inference performance, it would be interesting to investigate custom network architectures or available networks with smaller input dimensions, which are pre-trained similarly on a general purpose challenge (e.g. ImageNet Large scale Visual Recognition Challenge) and afterwards fine-tuned to the specific requirements of the RoboCup environment.

Furthermore the performance could possibly be enhanced by adapting models to the specific hardware capabilities. For example TensorRT[9] optimizes trained neural networks for execution on NVIDIA hardware by weight quantization (i.e. quantize floating point weights to 8-bit integers), layer and tensor fusion and other optimizations. However at the time of writing this framework is only available as a release candidate for the Jetson TX2 board currently missing some APIs when compared to the x86 version.

To improve the validation process, the ability to explain the network's decision may be useful. Selvaraju et al. developed Gradient-weighted Class Activation Mapping (Grad-CAM) [17], which highlights regions, that are 'important' for the prediction outcome. Further investigation into explanations of the

[9] https://developer.nvidia.com/tensorrt.

network's decision may allow to decide whether a network has actually learned the general concept of a soccer ball, or something else.

## 7 Conclusion

Ball detection by deep neural networks in the RoboCup humanoid league proved promising. The accuracy is adequate for the intended application, leading to less false negatives compared with previous approaches, as the Haar cascade algorithm that was used before. It is possible to adjust the approach to changing on-site conditions during a competition. By including a Jetson board in the robot, the use of CNNs is computationally feasible in a ROS based architecture.

The approach fosters the cooperation between teams by the possibility to build up common training sets and to share ROS nodes with pre-trained classifiers with other teams.

## References

1. Albani, D., Youssef, A., Suriani, V., Nardi, D., Bloisi, D.D.: A deep learning approach for object recognition with NAO soccer robots. In: Behnke, S., Sheh, R., Sarıel, S., Lee, D.D. (eds.) RoboCup 2016. LNCS (LNAI), vol. 9776, pp. 392–403. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68792-6_33
2. Balleda, K., Menon, S.K.: D-face: parallel implementation of CNN based face classifier using drone data on K40 & Jetson TK1 (2015)
3. Cruz, N., Lobos-Tsunekawa, K., Ruiz-del Solar, J.: Using convolutional neural networks in robots with limited computational resources: detecting NAO robots while playing soccer. arXiv preprint arXiv:1706.06702 (2017)
4. Eisenbach, M., Stricker, R., Seichter, D., Vorndran, A., Wengefeld, T., Gross, H.M.: Speeding up deep neural networks on the jetson TX1. In: CAPRI 2017, p. 11 (2017)
5. Gerndt, R., Seifert, D., Baltes, J.H., Sadeghnejad, S., Behnke, S.: Humanoid robots in soccer: robots versus humans in RoboCup 2050. IEEE Robot. Autom. Mag. **22**(3), 147–154 (2015)
6. Javadi, M., Azar, S.M., Azami, S., Ghidary, S.S., Sadeghnejad, S., Baltes, J.: Humanoid robot detection using deep learning: a speed-accuracy tradeoff. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) RoboCup 2017. LNCS (LNAI), vol. 11175, pp. 338–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_28
7. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
10. Lee, S., Son, K., Kim, H., Park, J.: Car plate recognition based on CNN using embedded system with GPU. In: 2017 10th International Conference on Human System Interactions (HSI), pp. 239–241. IEEE (2017)
11. Li, X., Lu, H., Xiong, D., Zhang, H., Zheng, Z.: A survey on visual perception for robocup msl soccer robots. Int. J. Adv. Rob. Syst. **10**(2), 110 (2013)

12. Mayer, G., Kaufmann, U., Kraetzschmar, G., Palm, G.: Neural robot detection in RoboCup. In: Wermter, S., Palm, G., Elshaw, M. (eds.) Biomimetic Neural Learning for Intelligent Robots. LNCS (LNAI), vol. 3575, pp. 349–361. Springer, Heidelberg (2005). https://doi.org/10.1007/11521082_21
13. Mhalla, A., Gazzah, S., Ben Amara, N.E., et al.: A faster R-CNN multi-object detector on a Nvidia Jetson TX1 embedded system. In: Proceedings of the 10th International Conference on Distributed Smart Camera, pp. 208–209. ACM (2016)
14. Militão, G., Colombini, E., Técnico-IC-PFG, R., de Graduação, P.F.: RoboCup soccer ball depth detection using convolutional neural networks (2017)
15. O'Keeffe, S., Villing, R.: A benchmark data set and evaluation of deep learning architectures for ball detection in the RoboCup SPL. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) RoboCup 2017. LNCS (LNAI), vol. 11175, pp. 398–409. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_33. Robocup symposium: poster presentation
16. Seki, Y., et al.: CIT Brains (kid size league) - team description paper (2017)
17. Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D.: Grad-CAM: why did you say that? arXiv preprint arXiv:1611.07450 (2016). https://arxiv.org/pdf/1611.07450.pdf
18. Speck, D., Barros, P., Weber, C., Wermter, S.: Ball localization for Robocup soccer using convolutional neural networks. In: Behnke, S., Sheh, R., Sarıel, S., Lee, D.D. (eds.) RoboCup 2016. LNCS (LNAI), vol. 9776, pp. 19–30. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68792-6_2
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. arXiv:1512.00567, December 2015
20. Taleghani, S., Aslani, S., Shiry, S.: Robust moving object detection from a moving video camera using neural network and kalman filter. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS (LNAI), vol. 5399, pp. 638–648. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02921-9_55
21. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, p. I. IEEE (2001)