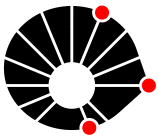
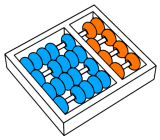




RoboCup Soccer Ball Depth Detection using Convolutional Neural Networks

G. Militão *E. Colombini*

Relatório Técnico - IC-PFG-17-06
Projeto Final de Graduação
2017 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Ball Depth Detection using CNN

Gabriel Militão Esther Colombini

Abstract

RoboCup, a soccer competition played by autonomous robots, raises a variety of hard problems in different fields such as Artificial Intelligence and Dynamics. In particular, vision is the essential input to take actions based on this highly dynamic environment and, although traditional techniques might be cheaper for object detection, new rules such as multicolored balls require more robust detection methods and allow a significant computing power improvement. Given these recent changes, this work presents a solution for ball localization using convolutional neural networks (CNNs), a largely applied machine learning technique holding the state of the art technology for several imagery problems. To perform the object detection, we train the CNN with robot's camera images as input and the ball's coordinates relative to the robot as output.

1 Introduction

Since RoboCup started on 1997 [1], teams have relied on traditional color segmentation and edge detection algorithms for the robot's vision [2, 3]. These methods are faster and easier to implement, not requiring data gathering and preprocessing for large datasets [4], a key factor in this interdisciplinary competition. Besides that, given the real-time necessity imposed by the game and the limited on-board computational resources enforced by the robot's embedded system, processing time is a critical constraint that also favors these algorithms.

However, in the past few years, the competition has been changing its rules towards more realistic game fields [5]. Allowing, for example, at most half of the ball to be formed by any combinations of colors and additionally switching from artificial light to natural light. These changes significantly impact traditional vision methods based on color histograms, since the ball's color distribution might vary based on angle and the environment's lightning. It pushes competitors to come up with new solutions for these problems, aligning with RoboCup's dream of, by the middle of the 21st century, a team of fully autonomous humanoid robot soccer players winning a soccer game against the winner of the most recent World Cup.

At the same time, recent advancements on deep learning allowed a paradigm shift in pattern recognition, from using handcrafted features together with statistical classifiers, to using general-purpose learning procedures to learn data-driven representations, features, and classifiers together. It has been specifically successful in vision problems, where it achieved the state of the art results for several problems [6]. These methods require moreover, more processing than traditional methods, becoming an issue for robotics' applications.

Inspired by this recent trend [7, 8], this work proposes a robust solution to the this more complex setting and starts to think about 2050 [9], facing this computing power challenge by adapting convolutional neural networks (CNNs) for RoboCup's ball localization.

In session 2, we describe the general theory behind CNNs, while section 3 goes deeper into the specifics of this experiment, ranging from the dataset description, to the proposed architecture and section lastly presents the results achieved and their discussion.

2 Convolutional Neural Networks

2.1 Overview

Neural Networks are a family of learning algorithms loosely inspired by the brain's neuron network. They have layers of nodes inter-connected by weighted edges, similarly to the architecture of neurons connected by synapses. The network has predefined functions that, parameterized with the edge's weights, takes an input an applies a sequence of functions to produce an output. In each inner layer, each node performs a dot product between its previous layer neighbors' output and the incoming edge weight, adding this value with its own bias, as demonstrate in Figure 1. Then, it takes this sum and applies a chosen *activation function* to introduce non-linearity to the model, forwarding the result to the next layer. In Neural Networks, this non-linearity is a key element because it allows the model to recognize more complex patterns and, differently from linear classifiers, can learn separately distinct variations of the same class.

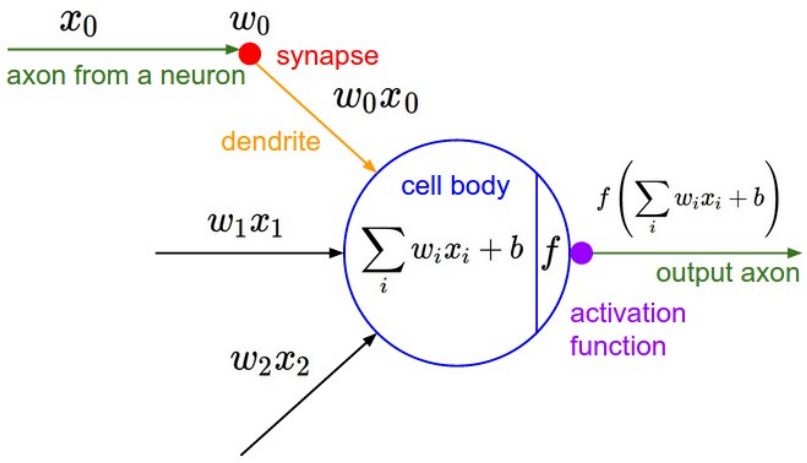


Figure 1: Artificial Neuron's representation, from [10]

During training, the network uses a suitable *cost function* to compare its output to the expected value. Its goal is to minimize this output cost adapting the network's parameters, the edge weights. In order to accomplish this task, the network uses *backpropagation*, an algorithm to compute the gradient of the cost function with respect to each weight. Knowing the gradient allows the neural network to adjust its parameters accordingly, rerun with the new parameters and decrease its cost. This training phase consists of several

iterations of this cost computing and backpropagation process optimization and, in case its well-designed, should converge to a minimum with a low cost. So, another similarity with their biological motivations can be observed from neural networks' behavior; they are feature learning methods, where no specific task is given, but the model learns and adapts based on expected output for each input.

2.2 Layers

A Convolutional Neural Network (CNN) differs from traditional Neural Networks by the existence of convolutional layers. These networks hold state of the art solutions for image classification [11, 6, 12] and object detection [13, 14, 15], and their architecture give a clue for the existence of these results. Firstly, unlike the normal version that requires an 1D conversion for the input, convolutional layers take multidimensional vectors, allowing spacial feature extraction. Additionally, although *fully connected* layers, i.e., layers where every node between sequential layers have edges between them, are ideal, they are impractical for a large number of nodes. So, convolutional layers are *locally connected*, where each neuron is connected to only a local region of the input and, although there's a loss in this design, in images, where spatial correlation is usually local, its impact on results is attenuated.

As illustrated in Figure 2, convolutional layers work like a sliding windows convolving through the input's width and height, computing dot products between the input and its parameters, the *kernel*. It produces a 2-dimensional activation map at every spatial location and, at each training iteration, learns filters that activate on visual features at these locations.

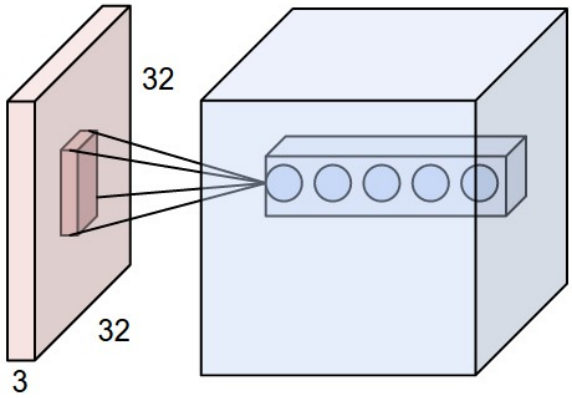


Figure 2: Convolutional Layer Representation, from [10]

After convolutional layers, it is common practice to insert Pooling layers in CNNs. This kind of layer reduces tensor spatial size, decreasing the number of parameters in the following layers and speeding up the networks output computation. The most utilized function for this layer is the max pooling, where a filter is applied spatially through it, getting the maximum value in that region. Figure 3 exemplifies a 2x2 filter max pooling layer applied to a tensor, downsampling its width and height by half.

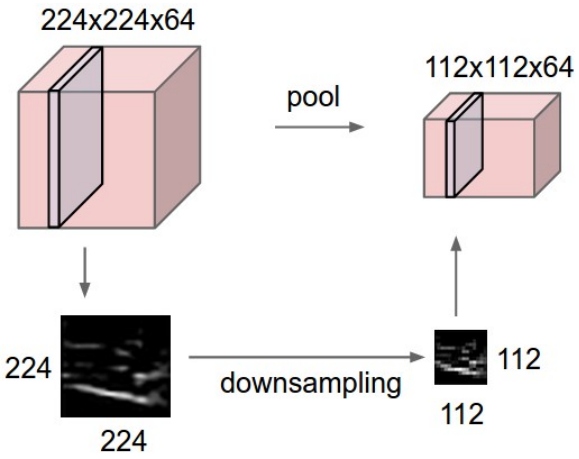


Figure 3: Pooling Layer Example, from [10]

Finally, after a sequence of convolutional and pooling layers, the network is smaller enough so that fully connected layers can be applied. These layers then produce the output according to the expected dimensions, preparing for cost computation.

3 Experiment

This experiment focused on a small network, with only a few layers, given the robots restraints previously mentioned. First, in 3.1, we discuss the dataset created for this work and in 3.2 present the design decisions for the network used. Later, we go deeper into the technical details on how the experiment was run and present the achieved results.

3.1 Dataset Description

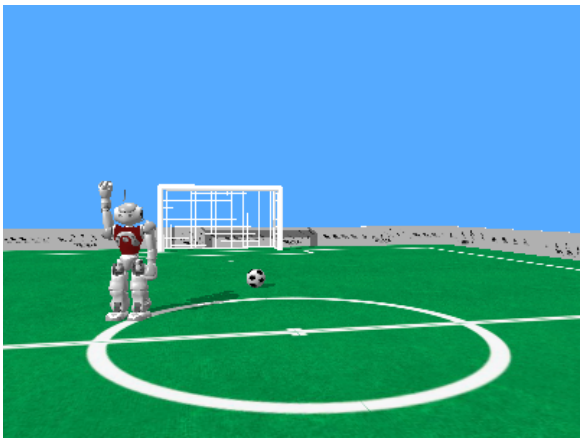


Figure 4: Dataset example image, taken from simulator's robot camera

As previously mentioned, CNNs need larges datasets [4]. For our experiment, we used Webots, a robotics simulator, to create RoboCup's environment [16]. As demonstrated in Figure 4, the ball was colored black and white and the field had an additional NAO robot. We took automated periodic snapshots from the robot's camera in different situations and the corresponding ball's relative position with respect to the robot. The ball appears at least partially in all images, in distances ranging from 1 to 5 meters. The dataset contains 400 images of dimension $400 \times 300 \times 3$ without any preprocessing, from which 350 were used for training and 50 for testing.

3.2 Proposed Architecture

Our experiment was based in a short version of deep neural networks in order to comply with the robot's constraints. So, as illustrated in Figure 5, it has only two convolutional layers followed by max pooling layers and two fully connected layers before generating the output.

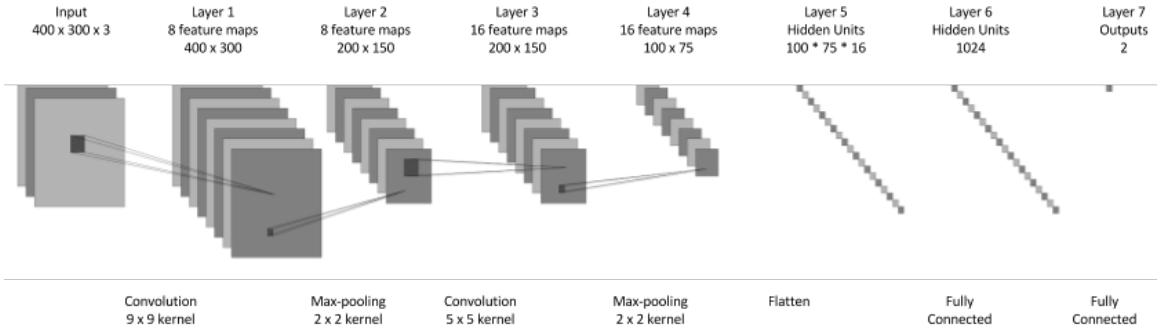


Figure 5: Convolutional Neural Network model

For the convolutional layers, we used kernels of size 9 and 5. The size drop is applied to adapt to the second convolution layer's dimension, resulted from max pooling that dropped the dimensions by half.

In all cases, the rectifier linear unit (ReLU) function, defined in Equation 1, is applied as the activation function. It's simpler, has higher efficiency and does not have problems such as the gradient vanishing one observed in other traditional functions such as sigmoid and tanh [17, 18].

$$h(x) = max(0, x) \tag{1}$$

A common problem faced in machine learning systems is *overfitting*, i.e., becoming too adapted to the training dataset and not to the desired general problem. In order to solve this problem, a couple of regularization techniques were applied. During training, fully connected layers randomly discard values according to a constant in each iteration, in a process known as *dropout*. It increases training loss but makes our model more general and thus reduces overfitting for the input. Another method applied is adding a function of the

network's weights as part of the cost function. The network then tries to minimize the weights, simplifying the model and reaching better results.

After the fully connected layers, we need to apply a suitable cost function for our problem. Since we are using the robot's coordinates as the ground truth, we compute the Euclidean distance between the predicted and truth coordinates, our measuring error, as stated in Equation 2. The cost will be then described as the mean of the relative error for this metric.

$$c(x) = \frac{1}{n} \sum^{n} \frac{dist(f(x_i) - y_i)}{dist(y_i)} \tag{2}$$

Once the cost is computed, backpropagation starts. To determine the next weights, an algorithm must be applied to solve the minimization problem. In this work, the stochastic gradient descent algorithm, that takes a step towards the gradient and recomputes the parameters, is used. This step is based on a hyperparameter called *learning rate*, which, if too big, can lead the network to diverge regions and in case is too small, can trap the parameters in local minima regions, with global high costs.

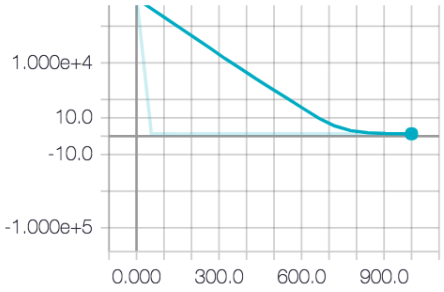
After the convolutional neural network's training process, we evaluate it against our test dataset. The metrics analyzed were error, as computed by Equation 2, and average absolute angle error from the predicted ball's location to its real position. The former metric can reveal the real detection's impact on RoboCup for larger distances, since the direction can be sufficient for the robot's decision taking.

Finally, to implement the network's code, we used Google's TensorFlow [19], an open source machine learning and deep neural networks framework. It provides the main layer implementations, functions and metrics optimized for running in both CPU and GPU, besides all the necessary structure to train and test these networks.

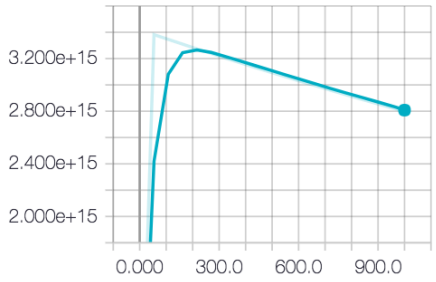
3.3 Results and Discussion

The Convolutional Neural Network was run for 1000 iterations, for a few tests. Regarding the learning rate, for a value of 0.0001, the network reached an average error of 90%. Increasing this hyperparameter to 0.01 decreased the error to 45%. As mentioned in Section 3.2, regularization techniques were applied in order to reduce overfitting, so the training cost shown in Figure 6b becomes several orders of magnitude greater than than the computed error in Figure 6a. Altough odd, this decision improved the final error to 38%.

Despite of the previously stated advancements, the obtained result is not enough for practical usage and one of the main factors for it was the dataset quality. First, it had only 400 images, an insufficient number for the CNN to detect the expected patterns. Besides that, the simulator is not the ideal source of images, constraining the diversity of situations on the acquired data and therefore limiting the CNN's ability to recognize the general problem. Additionally, as noted by the learning rate decision impact, hyperparameter tuning is very important in CNNs. So, dropout rates, learning rate and kernel sizes play a significant result of the final outcome. Lastly, weight initialization also has a powerful impact on the initial region to be optimized, where a bad initial decision can lead to bad results for any experiment. From a design perspective, another factor is the chosen output format.



(a) Graph of Log Error x Iteration



(b) Graph of Training Cost x Iteration

Even though they are linearly related, world coordinates instead of images coordinates might have affected CNNs ball's localization's effectiveness worse than expected.

References

[1] R. Federation, 2017, available at http://www.robocup.org; accessed July 05, 2017.

[2] R. Hanek, T. Schmitt, S. Buck, and M. Beetz, "Towards robocup without color labeling," *Lecture notes in computer science*, pp. 179–194, 2003.

[3] C. L. Murch, S. K. Chalup *et al.*, "Combining edge detection and colour segmentation in the four-legged league," in *Australasian Conference on Robotics and Automation (ACRA'2004)*, 2004.

[4] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 253–256.

[5] R. Team, "Robocup soccer humanoid league rules and setup," 2015, available at https://www.robocuphumanoid.org/wp-content/uploads/HumanoidLeagueRules2015_DRAFT_20141205-with-changes.pdf; accessed July 05, 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] D. Albani, A. Youssef, V. Suriani, D. Nardi, and D. Bloisi, "A deep learning approach for object recognition with nao soccer robots," in *Robocup Symposium: Poster presentation*, 2016.

[8] D. Speck, P. Barros, C. Weber, and S. Wermter, "Ball localization for robocup soccer using convolutional neural networks," in *Proceedings of the 2016 RoboCup International Symposium (RoboCup'2016)*, 2016.

[9] U. Frese, T. Laue, O. Birbach, and T. Röfer, "(a) vision for 2050-context-based image understanding for a human-robot soccer match," *Electronic Communications of the EASST*, vol. 62, 2013.

[10] Stanford University, "CS231n convolutional neural networks for visual recognition," 2017, available at http://cs231n.github.io/convolutional-networks/; accessed July 05, 2017.

[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[13] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2147–2154.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[15] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in neural information processing systems*, 2013, pp. 2553–2561.

[16] Cyberbotics, "Webots reference," 2017, available at https://www.cyberbotics.com/doc/reference/nodesand-api-functions; accessed July 05, 2017.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[18] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean *et al.*, "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf