



Toward Robotic Cognition by Means of Decision Tree of Deep Neural Networks Applied in a Humanoid Robot

Isaac J. Silva¹ · Claudio O. Vilão Junior¹ · Anna H. Realí Costa² · Reinaldo A. C. Bianchi¹

Received: 29 January 2020 / Revised: 10 February 2021 / Accepted: 17 March 2021 / Published online: 16 April 2021
© Brazilian Society for Automatics–SBA 2021

Abstract

One of the challenges of Deep Learning research is to develop algorithms for mobile robotic agents that operate in uncontrolled environments, in which dynamic changes and limited processing power are common restrictions. A common solution is to develop separate vision and decision modules, so that the former is based on deep neural network architectures and the latter is based on rules, and then interconnect them. The drawback of this solution is that the modules need to exchange high-level information about the objects in the scene, which are usually the positions of all objects in the scene, and this is computationally expensive. To address this problem, this paper presents a Decision Tree of Deep Neural Networks (DT-DNN) that aims to perform end to end—from image to decision—processing, and, thus, eliminating the need for quantitative and relational information about the image. This model is composed of smaller and more specialized modular DNNs, thus solving the trade-off between performance and inference time. Experiments were carried out using a real robot in the RoboCup Humanoid League domain in a soccer field, and also in simulation. We compared DT-DNN with several traditional DNN architectures. From the results, it is possible to conclude that the use of the DT-DNN made the system simpler and more robust, with fewer parameters to be adjusted, reducing the time spent with inference and also increasing the performance when compared to the traditional approach.

Keywords Cognitive robotics · Mobile robots · Robot learning · Robot vision systems · Deep neural networks

1 Introduction

Deep Learning techniques (LeCun et al., 2015) are a very active research area, as can be seen not only by a large number of works presented at conferences and specialized journals (Guo et al., 2016; Goyal & Benjamin, 2014; Gu et al., 2015; Schmidhuber, 2015) but also by the reported successful applications (Bojarski et al., 2016; Young et al., 2018; Deng, 2014; Heaton et al., 2016). Even achieving such results, the necessity of applying those techniques in non-laboratory environments (Ersen et al., 2017) is a major challenge for robotics researchers.

The RoboCup Challenge (Kitano et al., 1997) proposes soccer competitions between robots in non-laboratory environments, in order to promote research in Robotics and

Artificial Intelligence. Most teams participating in the competition have computer vision systems based on techniques such as thresholding, windowing, segmentation, or convolutional neural networks to find objects on the playing field, usually robots, field lines, and the ball. Even though convolutional neural networks are available for off-the-shelf use, many researchers to the robot soccer domain make adaptations or develop their own architectures (Speck et al., 2016; Team description paper, 2017; Javadi et al., 2017; Albani et al., 2016; Schneckeburger et al., 2017; van Dijk & Scheunemann, 2018). Teams also rely on the development of vision and decision modules separately and then interconnect both (Team description paper, 2017; Ha et al., 2011; Perico et al., 2014).

The main goal of this paper is to unify the vision and decision modules of a humanoid soccer playing robot using Deep Neural Networks to abstract what is being seen in the image through a qualitative representation of the context of the environment and then generate a single decision control. The main advantage of our proposal is that it does not need to locate each object in the image, i. e., it does not need to find

✉ Isaac J. Silva
isaacjesus@fei.edu.br

¹ Centro Universitário FEI, São Bernardo do Campo, SP, Brazil

² Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil

its exact position in the scene to decide which action should be performed by the robot.

In order for deep learning to be used in real robots, we propose to use a Decision Tree of Deep Neural Networks (DT-DNN), which combines several DNNs using a decision tree, allowing the multi-class decision problem to be decomposed into several binary or minors classification problems. The decision tree approach has already been used with other types of classifiers (Madzarov et al., 2009) and with neural networks (Drake & Packianather, 1998), yielding good results. We show that this approach can also be used with deep neural networks.

In this way, the contributions of this work consist of not only demonstrating that it is possible to develop a Cognitive System with Deep Neural Networks, but also presenting a development that uses the Decision Tree of DNNs. With that, we were able to address the trade-off between accuracy and inference time using a combination of the robot's visual and decision-making processes, enabling the use of the cognitive system proposed in the embedded robot computers.

The remainder of this paper is structured as follows: Sect. 2 depicts related work. Section 3 provides the basic concepts of deep neural networks and decision trees. Section 4 presents the proposed system. Section 5 describes the experiments that were conducted in a real robot and a simulator, and Sect. 6 provides our conclusions.

2 Related Work

In the RoboCup Humanoid Robot League community there are several works on vision systems using DNN for object detection (Speck et al., 2016; Cruz et al., 2017; Javadi et al., 2017; Schneckeburger et al., 2017; van Dijk & Scheunemann, 2018). These systems are responsible for detecting a ball, robots and landmarks such as goal posts, line corners and circle-penalty points.

Javadi et al. (2017) performed training and testing of three different DNN architectures, LeNet, GoogLeNet and SqueezeNet, in order to classify the images into humanoid robots images or non-robot images.

In order to detect a robot in the image, the algorithm has two stages: the first defines regions of interest (ROI) and, since these ROIs may contain false positives, the second stage checks once again whether the ROI images are images of humanoid robots.

The work of Speck et al. (2018) presents two DNN architectures to identify a ball inside a frame. The first architecture has three convolutional layers and two fully-connected layers to inform the ball location by two output layers representing the x- and y-coordinates. The second architecture is a Fully Convolutional Neural Network (FCNN) and the output is a heat-map image. According to the authors the ball position

has to be manually informed for each image in the training phase (Speck et al., 2016).

Schneckeburger et al. (2017) proposed the use of a Fully Convolutional Neural Network (FCNN) on the vision system of their adult size robot, named Sweaty, that recognizes and locates objects in images coming from its camera. In order to find a good architecture for the FCNN, they proposed three similar architectures, SweatyNet 1, 2 and 3, where SweatyNet-2 and SweatyNet-3 are variations of SweatyNet-1, with fewer layers and parameters. Their experiments were conducted on an Intel i7-3.5Ghz-CPU, and obtained a high hit rate in the detection of ball, field lines and goalposts. However, their solution still has problems with the location hit rate and the detection of opponents.

When evaluating the works published in the field of robot soccer, it is possible to notice that none of them uses DNNs to perform actions directly from the image acquired by the humanoid robot; in all published works, the DNN is used only for object detection and not for decision.

Our proposal, therefore, takes inspiration from a paper that was published outside the RoboCup domain: the work of Giusti et al. (2016), in which the authors have developed a DNN using a Semi-direct monocular Visual Odometry (SVO) to control a drone moving through forest tracks. The input of the DNN consisted only of RGB images acquired by a camera positioned on the front of the drone.

The DNN input image was reduced to 101×101 pixels. The network input is $101 \times 101 \times 3$ (width, height, RGB) followed by six hidden layers and three neurons in the output layer. The network output is composed of three classes: left, right and center. Using the DNN and the SVO, a reactive control was implemented, being able to control the drone's rotations and forward movement. Their work showed that a DNN can be used to decide which actions the drone should take, based only on images, similar to that used in the robot soccer domain. A drawback of their proposal is that the training images were taken by GoPro cameras, different from those used on the drone, leading to classification errors and resulting in lower performance in the classification made with the drone's camera (Giusti et al., 2016). Another difficulty reported in this work is the great variation that occurs among the images due to several factors, such as: lighting conditions, vegetation types, altitude, local topography, and many other factors. So there was a need to include a large variety of images in the training set (Giusti et al., 2016).

Our proposal seeks to circumvent these problems by using the same image capture systems in the training and execution phases, in addition to seeking during the training to expose the robot to the most similar conditions possible to those that it will encounter in the execution phase.

There is a large number of works on learning to control for different types of robots, such as Loquercio et al. (2018), where a eight-layers convolutional neural network is used to

learn how to safely drive a drone through the streets of a city, using data collected by cars and bicycles; Codevilla et al. (2018), that used imitation deep learning on high-level command input to learn to drive following roads and avoiding obstacles; and Rhinehart et al. (2019), that used an imitative learning approach in a dynamic simulated autonomous driving task.

Finally, although not applied in the robotic mobile agent domain, there is a growing number of research on the multi-class decomposition approach for Deep Learning: Roy et al. (2020) proposed an adaptive hierarchical network structure composed of deep convolutional networks that grows in a tree-like fashion to accommodate new classes of data while learning, preserving the ability to distinguish the previously trained classes; Kowsari et al. (2017) also followed the hierarchical approach to large document collections, employing stacks of deep learning architectures to provide specialized understanding at each level of the document hierarchy; and Yan et al. (2015) decomposed the image classification problem by embedding deep CNNs into a category hierarchy.

3 Theoretical Background

In this section we provide the fundamental concepts of deep neural network and decision trees, which are the basis of our proposal.

3.1 Deep Neural Network

A Deep Neural Network (DNN) is a kind of Artificial Neural Network (ANN) with a higher level of complexity, having a large number of layers between the input and output of the network. there is no consensus on how many layers a DNN must have, but the more layers it has, the deeper the network is considered.

A typical type of DNN is a Convolutional Neural Network (CNN), which has been widely applied to image data. This type of DNN is composed of several layers consisting of three stages: parallel convolutions, applied in order to make a series of linear activation; each linear activation run through a nonlinear activation function; a pooling function, that are used by almost all convolutional networks, is applied to replace the output of the layer, aiming to make the representation become invariant to small translations (Goodfellow et al., 2016). Finally, the Dropout (Hinton et al., 2012) technique is applied during the training in fully-connected layers to reduce over-fitting.

The training of these networks can be performed by using the back-propagation method, or other gradient learning, such as the Stochastic Gradient Descent (SGD) or the Adam Algorithm (Kingma & Ba, 2014).

The current popularity of DNN started in 2012, when the SuperVision team of Toronto's University employed a DNN called AlexNet (Krizhevsky et al., 2012) in the Imagenet Large Scale Visual Recognition Challenge (ILSVRC),¹ a well-known computer vision competition where algorithms have to classify images from a set of more than 1 million images in 1000 different classes. The AlexNet became known that year for presenting the best performance in the competition, achieving a top-5 error rate of 15.315%, while the second best team presented in the same category a rate of 26.172%.

In 2014, Google DeepMind researchers presented a DNN called GoogLeNet, winning the ILSVRC 2014 competition with a top-5 error rate of 6.67%. GoogLeNet is a DNN of 22 layers proposed by (Szegedy et al., 2015). In 2015, Microsoft team (MSRA) presented a DNN called ResNet, winning the ILSVRC 2015 classification competition with a top-5 error rate of 3.57%. And recently the DNN called SENet won the ILSVRC 2017 classification competition with a top-5 error rate of 2.25%, which is better than the human result.

Finally, recent research have constantly improved the performance while also trying to reduce the computational costs of this kind of classifiers (Sze et al., 2017; Carmichael et al., 2019).

3.2 Decision Trees

Decision Trees (DT) are one of the simplest forms of machine learning method, and until the 1980s, the decision tree was the main technique used in easy decision problems (Russell & Norvig, 2010). It is a supervised learning method used for classification and regression, which learns a set of if-then-else decision rules from data values, and handle both categorical and numerical data.

A decision tree is incrementally developed by creating decision nodes that splits a data set into smaller subsets. These nodes are arranged in the form of a tree with nodes and leaves. The deeper the tree, the more complex the decision model.

For decision making, a sequence of tests with these attribute values are performed. These tests are performed by the nodes of the tree, so each node decides to which branch the values will be routed to the next decision node, finally, the leaf nodes of the tree define the decision that should be made for such attribute values (Russell & Norvig, 2010). As output, we have a single value referring to the decision for a set attributes.

Finally, while learning a decision tree is a computationally expensive task, using one is not. Decision trees are constructed from several examples of the problem, and depending on it, the tree may be complex, and yet accurate

¹ <http://www.image-net.org/challenges/LSVRC>.



Fig. 1 Two humanoid robots used in the experiments: the Goalkeeper (on the goal line) and the Defender

and efficient (Quinlan, 1987). The efficiency of the Decision Trees comes from the fact that it bisect the space into smaller and smaller regions, whereas other classifiers, such as a neural networks, needs to compute the relation of the higher-dimensional data to several hyperplanes to be able to classify one sample: using a DT to make a decision corresponds to following a single branch of the tree, using simple binary classifiers, from the root to the leave, which is very fast.

4 The Proposed Algorithm

The proposal of this work is to develop a system that is capable of controlling a humanoid robot, unifying the vision and decision modules, by using a Deep Neural Network that receives the image from the robot's camera and sends a control action to the robot.

To be able to achieve the performance required by real robots, reducing the time spent during inference without losing precision, we propose to use a Decision Tree of Deep Neural Networks (DT-DNN). We applied the DT-DNN in the control of a humanoid goalkeeper robot (Figure 1). The goalkeeper must remain positioned in the goal area and check if there is any ball near it. As soon as the goalkeeper detects that the ball is close, it must go towards the ball and kick it away from the goal, and then return to the goal area.

The DT-DNN receives an image from the robot's camera and classify the image in a way that the classification result will be the action that the robot should take. Seven classes were created regarding the actions the robot should take when seeing one type of image:

- if seeing ball left \Rightarrow turn left;
- if seeing ball right \Rightarrow turn right;
- if seeing ball center \Rightarrow walk forward;
- if seeing kick right \Rightarrow kick right;

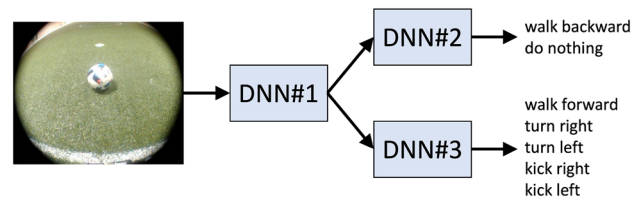


Fig. 2 The Decision Tree of DNNs used to control the goalkeeper

- if seeing kick left \Rightarrow kick left;
- if seeing no ball (empty field) \Rightarrow walk backward;
- if at goal \Rightarrow do nothing.

These actions must be taken by the DT-DNN, but the control process is responsible for performing these actions, so the DT-DNN controls the robot by deciding which actions the control process should perform.

The main contribution of this work is the development of the Decision Tree of DNNs. With the Decision Tree of DNNs it was possible to decrease the inference time and increase the accuracy, so it became possible to use the cognitive system with Deep Neural Networks on the computers embedded in the robots.

To decrease the inference time of a DNN, it is possible to decrease the number of neurons in the networks, however, this decrease also affects accuracy in a negative way. This work shows that the Decision Tree of DNNs allows the number of neurons in the network to be reduced, and that even decreasing the number of neurons, there are increases in the accuracy of the system. That's because instead of using a single network that classifies everything, we divide it into small networks that perform part of the classification.

The Decision Tree of DNNs used to control the goalkeeper is composed of 3 DNNs as shows Fig. 2, where the first DNN receives the image of the robot camera and performs a binary classification, deciding whether there is a ball in the image. If there is no ball, the image is forwarded for the second network (DNN#2 in Fig. 2) that classifies whether the robot is positioned at the goal or not, and decide between *doing nothing* or *walking backwards*, respectively. If there is a ball in the image, it is forwarded to the third DNN (DNN#3 in Fig. 2) that classifies the image of the ball between 5 possible classes, and informs the network output regarding the action to be taken. The three DNNs used (DNN#1, DNN#2 and DNN#3 in Fig. 2) have the same 5-layer architecture, consisting of three convolutional layers, the first one using max-pooling, followed by two fully connected layers, described in Table 1.

We also propose a method to reduce the classification error in image sequences. Consider a series of classifications carried out in such a way that all past classifications indicate the correct class, except the C_t classification, i.e., $C_{t-N} = \dots = C_{t-2} = C_{t-1} \neq C_t$, where C_t is a mis-

Table 1 The 5-layer architecture used for each DNN in the Decision Tree

Type (layer)	Kernel	Stride	Pad	Filters
Convolutional (1)	8×8	4	0	32
Max-pooling (1)	3×3	1	0	32
Convolutional (2)	4×4	2	0	64
Convolutional (3)	3×3	1	1	64
Fully-connected (4)	–	–	–	512
Fully-connected (5)	–	–	–	7

classified class and the others $C_{t-N} \dots C_{t-1}$ are correct classifications.

In order to avoid errors like in C_t that can occur between classifications and thus prevent the robot from performing an incorrect action, we propose a new method to decide which action should actually be taken.

In the DNN used in this work, the model outputs are represented by a probability distribution using Softmax, where each class has a probability Pr related to the resulting inference. Softmax calculates a probability for each class, with the sum of the probabilities being exactly 1. Softmax (Goodfellow et al., 2016) is defined by:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}, \quad (1)$$

and the output will be represented by a value within the range 0 to 1, where: z_i is the output value of neuron i and k is the number of outputs. Based on that, we calculate the sum of the last class probabilities using a discount factor,

$$\text{sumPr}(c) = \sum_{i=0}^N \gamma^i \text{Pr}_{t-i}(c), \quad (2)$$

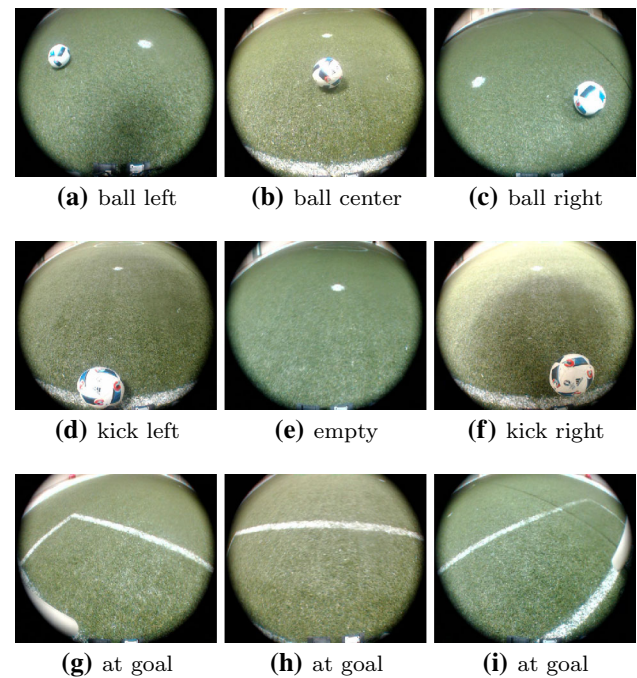
where Pr is the class probability given by the resulting inference from the DNN, computed using a Softmax function, c is the class, γ is the temporal discount rate, and N is the number used to calculate the last class probabilities. In all experiments, $\gamma = 0.9$ and $N = 10$ were used.

Thus, the action that should be taken is the one that presents the highest $\text{sumPr}(c)$:

$$C = \arg \max_c \text{sumPr}(c), \quad (3)$$

where C is the selected class.

The training of the network was performed using images related to each decision context, illustrated in Fig. 3. The images were acquired by the robot camera in random positions in the field. Several images were labeled in classes regarding the action that the robot must perform when viewing similar situations in the current image.

**Fig. 3** Sample of images used to train each decision class

5 Experiments

This section presents three experiments that were executed to test our proposal: using a single DNN controlling a real robot; using the DT-DNN also on the real robot; and experiments performed in a robotic simulator.

The humanoid robots used in these experiments were developed by University Center FEI (Perico et al., 2014), and have their mechanics based on the DARwIn-OP (Ha et al., 2011), thus it uses the same gait pattern generator. The robots have 49 cm of height, 3 kg of weight, and 20 Dynamixel MX-28 servo-motors. As sensors, it uses an UM7 Ultra-Miniature Orientation Sensor and a Logitech HD Pro Webcam C920 (Full HD). The computer used as the main processing unit is an Intel NUC² Core i5, with 8GB of RAM memory and 120GB SSD. In order to increase the field of view of the camera and eliminate the need of using pan and tilt servo motors in the head to look for the ball, a fish-eye lens was added to the robot's camera.

The Humanoid robot has an architecture named Cross Architecture (Perico et al., 2014), which has completely independent processes:

Vision process

that is responsible for object detection—this process must be able to find the ball, the goals, the field lines, the other

² <http://www.intel.com/content/www/us/en/nuc/overview.html>.

Decision process	robots and inform the distances among the robot and the other objects; that must be able to consolidate all the information received from the Vision, Localization and Communication processes and uses these data to make the decision;
Localization process	that must be able to know its position in the soccer field, as well as its direction;
Communication process	that must be able to manage the wireless communication to receive and transfer information among the robots and also to control the messages received from the referee;
Control process	that is in charge of controlling all the servomotors of the body—it is responsible for making the robot walk, turn around, stand up, kick the ball and so on, for this, the control is composed by several processes responsible for performing the inverse kinematics, the gait generator, reading and writing in the motors and so on;
Management process	that is used to launch, synchronize and monitor all the processes.

In the experiments with the real robot, a white ball was used, in a turf soccer field of 9×6 meters, with white goal posts, as described in the RoboCup Humanoid League Rules (RoboCup Federation, 2020). Experiments made in simulation used the Cyberbotics Webots Open Source Robot Simulator,³ where the DARwIn-OP robot (Ha et al., 2011) is modeled.

For all the experiments, training was performed in an Intel i7 2.8GHz computer with 32GB of memory, 480GB SSD, GPU NVIDIA GeForce GTX 1060 with 6GB of RAM memory, using Linux Ubuntu 14.04 and 16.04. The algorithms were implemented using the C++ and Python programming languages, and the DNNs were implemented using Caffe.⁴

To evaluate the performance of the algorithms we used the Confusion Matrix, Precision and Recall metrics (Ting, 2010) and also the time to perform the computation. The Confusion Matrix is a table layout created for easy visualization of the

results of a classification, where each row displays the number of instances in a predicted class while each column shows the number of instances in an actual class. From the Confusion Matrix it is easy to compute the Precision and Recall (also known as Sensitivity or Hit-rate) of a classification. Precision is defined as the number of true positives (TP) over the number of true positives (TP) plus the number of false positives (FP),

$$Precision = \frac{TP}{TP + FP}, \quad (4)$$

while Recall is defined as the number of true positives (TP) over the number of true positives (TP) plus the number of false negatives (FN),

$$Recall = \frac{TP}{TP + FN}. \quad (5)$$

5.1 Single DNN Experiments Performed on the Real Robot

In order to test the performance of the end-to-end system using only one DNN, we implemented four different architectures, ranging from a very simple one to more complex architectures. The tested architectures used are:

- *Conv4* a simple convolutional DNN with 4 convolutional layers and 3 fully connected layers. The ReLu nonlinear activation function was used in all the layers, and dropout was used in the first two fully-connected layers during training (Table 2). This architecture was proposed as a way to reduce inference time, as the limitations in computational power of the real robots may cause more complex architectures to have unacceptable decision processing time. This architecture was used with two different input sizes, 110×110 and 256×256 pixels.
- *AlexNet* Krizhevsky et al. (2012) a well-known convolutional DNN with 5 convolutional layers connected to overlapping max-pooling layers and 3 fully connected layers. As in the Conv4, ReLu activation function was used in all the layers, and dropout was used during training. This architecture was used with two different input sizes, 110×110 and 256×256 pixels.
- *SqueezeNet* Iandola et al. (2016) a convolutional DNN that is 18 layers deep, but uses less parameters than AlexNet, with the same level of accuracy. This architecture was used only with 256×256 input.
- *GoogLeNet* Szegedy et al. (2015) another traditional off-the-shelf network, it is 22 layers deep. This architecture was used with only one input size, 256×256 pixels.

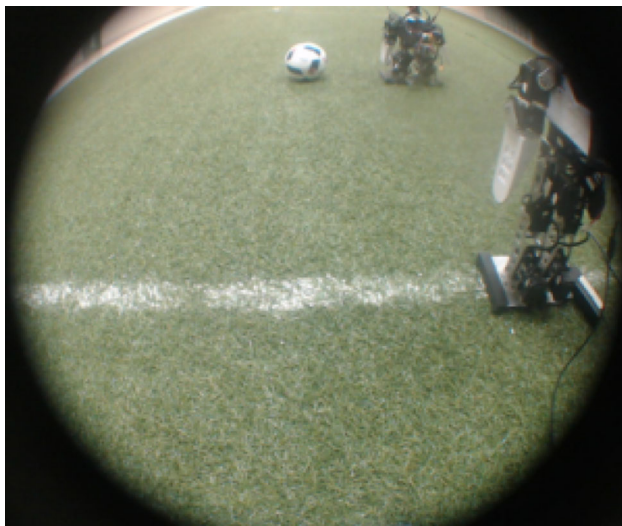
The DNN parameters were configured with the following values: the learning rate was initialized at 0.01; In the Digits

³ <https://cyberbotics.com>.

⁴ <http://caffe.berkeleyvision.org/>.

Table 2 The Conv4 network architecture

Type (layer)	Kernel	Stride	Pad	Filters
Convolutional (1)	11×11	4	0	96
Max-pooling (1)	3×3	2	0	96
Convolutional (2)	5×5	1	2	256
Max-pooling (2)	3×3	2	0	256
Convolutional (3)	3×3	1	1	384
Convolutional (4)	3×3	1	1	256
Max-pooling (4)	3×3	2	0	256
Fully-connected (5)	–	–	–	512
Fully-connected (6)	–	–	–	512
Fully-connected (7)	–	–	–	7

**Fig. 4** Image with ball and other robot

Nvidia the crop size is a data transformations parameter that can be used in training to resize the image, but was not used in this work, then the Crop Size is none; During the experiments, the parameters of Eq. (2) were set as follows: $\gamma = 0.9$ and $N = 10$. The solver type chosen was Stochastic Gradient Descent with a batch size of 128 examples.

The dataset used consists of images extracted from the robot's camera, with the robot standing in the soccer field. In part of the images only the field and the ball can be seen (16,388 training + 5460 validation images), while in other images ball and robots positioned in random field positions are seen (2090 training + 522 validation images, see example shown in Fig. 4). The dataset also includes 1400 test images (700 with and 700 without robots on the images). The images have a dimension of 1280×720 pixels (RGB) without any preprocessing.

Table 3 shows the recall of each network applied in the test images, as well as the time spent to classify an image on an Intel NUC. Samples of classification time were averaged over

Table 3 DNN Recall and time to classify an image

Network	Input	Recall [%]	Time [s]	
			Mean	σ
Conv4	110×110	74.43	0.093	0.00602
AlexNet	110×110	77.00	0.131	0.00811
Conv4	256×256	83.86	0.458	0.02784
AlexNet	256×256	87.86	0.470	0.01591
SqueezeNet	256×256	79.57	0.139	0.01920
GoogLeNet	256×256	96.00	0.682	0.03386

10 min. From this table we can conclude that it is possible to run a DNN on an Intel NUC computer onboard the humanoid robot by reducing the input image size and number of neurons in the network, resulting in a tiny decrease in recall: the higher the recall, the longer the inference time.

Another test was performed whereupon the robot goalkeeper was controlled by the DNN.⁵ The robot goalkeeper always started positioned in the goal area, then the DNN process received the robot's camera image and responded with the action that should be taken for that image. During that test, the robot performed the actions expected it to take.

5.2 Decision Tree of DNNs

In the previous subsection we showed that it is possible to use a DNN to control one robot according to the proposal described in Sect. 4, however the higher the network recall, the longer is the inference time. In this section we will show that it is possible to improve the performance using a Decision Tree of DNNs described in Sect. 4 and compare its results with those obtained in the previous subsection.

5.3 Experiments Using a Robotic Simulator

This section shows the results of the proposal (Sect. 4) being applied in a robotic simulator.

In this experiment the Cyberobotics Webots simulator was used, a robotic simulator that offers a three-dimensional virtual environment with physical properties. The robot used in the experiments was Darwin-OP humanoid robot (Ha et al., 2011), located in a soccer field previously available in the simulator.

To acquire the images for the simulated robot data set, a number of videos were recorded and had their frames extracted. Videos were recorded from the robot's camera for each class (ball center, ball right, and others). Having separate videos makes it easier to extract the video frames directly for each specific class, avoiding the need to analyze and separate

⁵ Video available at <https://youtu.be/b3tebN9dwN8>.

Table 4 Recall and dispersion among classes for all networks tested in simulation

Network	Input	Recall [%]						
		ball center	ball left	ball right	kick left	kick right	at goal	empty
Conv4	110 × 110	100	100	99.8	100	100	100	100
AlexNet	110 × 110	100	99.8	99.8	100	100	100	100
Conv4	256 × 256	100	100	99.9	96.3	98.4	100	100
AlexNet	256 × 256	100	99.9	99.9	99.8	99.6	100	100
SqueezeNet	256 × 256	99.9	100	99.6	99.5	99.3	100	100
GoogLeNet	256 × 256	100	99.8	100	100	100	100	99.7
DT-DNN	110 × 110	99.9	99.9	99.9	99.9	99.9	100	100

each frame into classes. To generate the videos, the simulator placed the robot and the ball in random field positions, while respecting the limit of each class.

The simulation image acquisition rate reaches 30 fps when running on a desktop computer equipped with a GPU, however the robot embedded system has limited resources. Hence, in order to resemble the image acquisition simulation with real robot image acquisition, it is necessary to apply a correction rate of 8 Hz (125 ms). The argument behind this choice is that this value seems to be suitable and appropriate for the robot to classify an image and take an action.

The generated dataset contains 36,644 training images, 12,214 validation images and 7000 test images, with resolution 640 × 360 × 3 (width, height, RGB-channels). The robot's camera in the simulation was set with a field of view of 2.3415 radians, to simulate a camera with a fisheye lens with 130 degrees of field of view.

Table 4 shows that all DNNs presented a recall of 100% or close to 100% in several classes. This high recall occurs because in the simulator images there are few variations such as illumination, and also the dataset in the simulator is larger than the dataset of the experiments in the real robot due to ease of capturing images.

5.3.1 Experiments Using a Real Robot

This section shows the results of the proposal (Sect. 4) being applied in a real robot.

Table 5 shows the recall of the Conv4, GoogLeNet and the DT-DNN networks applied in the test images, as well as the time spent to classify an image on an Intel NUC. It can be seen that the DT-DNN performs better than the best multi-class DNN presented in the first experiment, the GoogLeNet, spending less time than the simplest of those networks.

Tables 6, 7 and 8 show the confusion matrix of each DNN in the Decision Tree of DNNs: Table 6 presents the confusion matrix for DNN#1, the binary classifier for the presence of the ball; Table 7 shows the confusion matrix for DNN#2, the binary classifier for being at the goal; and Table 8 presents the confusion matrix for DNN#3, the classifier used then the

Table 5 DNN Recall and time to classify an image using DT-DNN

Network	Input	Recall [%]	Time [s]	
			Mean	σ
Conv4	110 × 110	74.43	0.093	0.00602
GoogLeNet	256 × 256	96.00	0.682	0.03386
DT-DNN	110 × 110	97.00	0.035	0.00216

Table 6 Confusion matrix for DNN#1 of the DT-DNN: binary classifier for the presence of the ball (output two classes: ball and no ball)

	Ball	No ball	Per-class recall
Ball	489	11	97.8%
No ball	0	200	100%

Table 7 Confusion matrix for DNN#2 of the DT-DNN: binary classifier for being at the goal

	At goal	Empty	Per-class recall
At goal	96	4	96.0%
Empty	0	100	100%

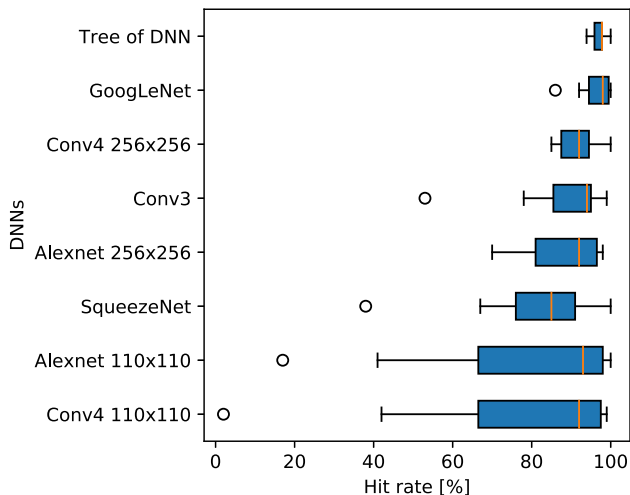
robot is with the ball. These Tables demonstrate that DNNs with fewer classes presented a recall per class greater than the using a single DNN to classify all 7 classes.

Figure 5 presents a box plot of the recall among classes for each DNN, the graph shows quartiles represented by blue rectangles (Q1 and Q3 at the ends of the rectangle and Q2 represented by an orange dash), the minimum and maximum value represented by a dash, and the outliers represented by a circle in the graph. Figure 5 shows that using DT-DNN, the dispersion of the hit rate is the smallest and it does not have outliers.

Table 8 shows that the class *ball center* was the one that presented the worst recall, because in the images of the ball on the right or ball on the left many times the ball was positioned in the threshold between the two classes, but this is not a problem when the network is controlling the robot, because

Table 8 Confusion matrix for DNN#3 of the DT-DNN: classify images with ball

	Ball center	Ball left	Ball right	Kick left	Kick right	Per-class recall
Ball center	96	2	2	0	0	96.0%
Ball left	0	100	0	0	0	100%
Ball right	0	2	98	0	0	98.0%
Kick left	0	0	0	100	0	100%
Kick right	0	0	0	0	100	100%

**Fig. 5** Recall and dispersion among classes for all networks tested

the robot going forward or turning a little on that threshold does not divert it from reaching the ball.

Table 9 shows the results for the Decision Tree of DNNs compared with the results of using only one multiclass DNN presented in the first experiment. In this table it is possible to observe that all classes of the Decision Tree of DNNs had an approximate recall of 100% with the lowest recall in the ball center class with 93.8%, different from the other DNNs.

Another test was performed whereupon the robot goalkeeper was controlled by the DNN.⁶ The robot goalkeeper always started positioned in the goal area and the DNN process received the robot's camera frame and responded with the action that should be taken for that image. During that test, the robot performed the actions we expected it to take.

In the experiments presented, the DNN architectures AlexNet, SqueezeNet, GoogLeNet and some architectures with 4 and 3 convolutional layers were used, because these types of architectures present an inference time of maximum 600 ms (GoogLeNet presents an inference time of approximately 600 ms) on an Intel NUC computer. Architectures

that have a time greater than 600 ms are infeasible for this project due to the dynamism of the game. During the gait, the humanoid robot was configured to perform one footstep every 600 ms, and a half-cycle (half footsteps) is 300 ms. Therefore, the decision taking should be less than 300 ms, in order to the decision have an effect so that the robot can change its direction during the gait.

Figure 6 already shows the number of network parameters, the sizes of the circles are proportional to the number of the respective network's parameters, and the number of parameters ranges from 5 million to 155 million parameters, represented by the colored circles to be used as a comparison with the inference time and the hit rate.

In order to facilitate the replication of these results and encourage the development of similar approaches by other researchers, all the software and data set used in this paper are available for download at: <https://github.com/Isaac25silva/Goalkeeper-DNN.git>.

6 Conclusions

This paper showed that a DNN can control a robot in a way that combines the processes of vision and decision, using only raw image from the robot's camera without any image preprocessing, guiding the control process to what action the robot must take to the image observed at that instant time.

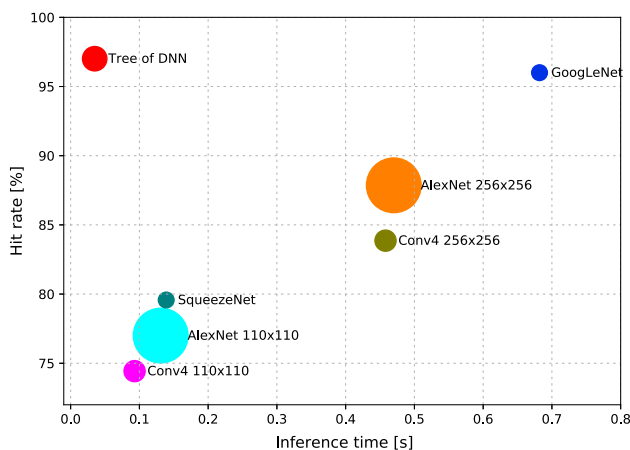
An advantage of this model is that, during training, there is no need to inform the precise location and boundaries of the object in the image, unlike other techniques, in which during training it is necessary to manually inform the object's position in the image.

This paper also showed that the use of a decision tree of DNNs reduced the inference time and also increased the recall, when compared to the use of a single multiclass DNN. In the experiments carried out, DT-DNN presented a better recall than the GoogLeNet, but with an inference time twenty times shorter, proving it is possible to obtain a high recall with a low inference time.

⁶ Video available at <https://youtu.be/b3tebN9dwN8#t=03m16s>.

Table 9 Recall and dispersion among classes for all networks tested

Network	Input	Recall [%]						
		Ball center	Ball left	Ball right	Kick left	Kick right	At goal	Empty
Conv4	110 × 110	92	02	91	98	42	97	99
AlexNet	110 × 110	92	17	93	99	41	97	100
Conv4	256 × 256	94	95	89	86	92	85	100
AlexNet	256 × 256	98	70	71	98	91	92	95
SqueezeNet	256 × 256	95	87	85	67	38	85	100
GoogLeNet	256 × 256	100	97	92	100	86	98	99
DT-DNN	110 × 110	93.8	97.8	95.8	97.8	97.8	96.0	100

**Fig. 6** Recall and inference time and number of network parameters for all networks tested in simulation

As a future work, our intention is to apply the model to other robot player roles such as attackers, who need to learn when to kick the ball towards the goal.

Acknowledgements The authors acknowledge the São Paulo Research Foundation (FAPESP Grants 2019/07665-4 and 2017/17291-9) and CNPq (Grants 307027/2017-1 and 425860/2016-7) for supporting this project. The authors acknowledge the support of NVIDIA Corporation with the donation of a Jetson kit. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES) - Finance Code 001.

References

- Albani, D., Youssef, A., Suriani, V., Nardi, D., Bloisi, D.D. (2016). A deep learning approach for object recognition with nao soccer robots. In *Robot World Cup*, pp. 392–403. Springer.
- Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016). End to end learning for self-driving cars.
- Carmichael, Z., Langroudi, H.F., Khazanov, C., Lillie, J., Gustafson, J.L., Kudithipudi, D. (2019). Performance-efficiency trade-off of low-precision numerical formats in deep neural networks. In *Proceedings of the conference for next generation arithmetic 2019*.
- Codevilla, F., Muller, M., Lopez, A., Koltun, V., Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*.
- Cruz, N., Lobos-Tsunekawa, K., Ruiz-del Solar, J. (2017). Using convolutional neural networks in robots with limited computational resources: Detecting nao robots while playing soccer. *arXiv preprint arXiv:1706.06702*.
- Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 1–29.
- Drake, P. R., & Packianather, M. S. (1998). A decision tree of neural networks for classifying images of wood veneer. *The International Journal of Advanced Manufacturing Technology*, 14(4), 280–285.
- Ersen, M., Oztup, E., & Sariel, S. (2017). Cognition-enabled robot manipulation in human environments: Requirements, recent work, and open problems. *IEEE Robotics & Automation Magazine*.
- Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., et al. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2), 661–667.
- Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Goyal, S., & Benjamin, P. (2014). *A survey*. CoRR: Object recognition using deep neural networks ([arXiv:1412.3684](https://arxiv.org/abs/1412.3684)).
- Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G. (2015). Recent advances in convolutional neural networks. *CoRR*, [arXiv:1512.07108](https://arxiv.org/abs/1512.07108).
- Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27–48. Recent Developments on Deep Big Vision.
- Ha, I., Tamura, Y., Asama, H., Han, J., & Hong, D. W. (2011). Development of open humanoid platform darwin-op. *IEEE in Proceedings of SICE Annual Conference (SICE)*, pp. 2178–2181.
- Heaton, J.B., Polson, N.G., Witte, J.H. (2016). Deep learning in finance. *ArXiv*, [arXiv:1602.06561](https://arxiv.org/abs/1602.06561).
- Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.
- Javadi, M., Azar, S.M., Azami, S., Ghidary, S.S., Sadeghnejad, S., Baltes, J. (2017). Humanoid robot detection using deep learning: a speed-accuracy tradeoff. In *Robot World Cup*, pp. 338–349. Springer.

- Javadi, M., Azar, S.M., Azami, S., Shiry, S., Ghidary, S.S., Baltes, J. (2017). Humanoid robot detection using deep learning: A speed-accuracy tradeoff. In *Robot World Cup*, pp. 338–349. Springer.
- Kingma, D., Ba, J. (2014). A method for stochastic optimization. arXiv preprint: Adam (arXiv:1412.6980).
- Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E. (1997). Robocup: The robot world cup initiative. In *Proceedings of the first international conference on autonomous agents*, AGENTS '97, pp. 340–347, New York, NY, USA. ACM.
- Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K. Jafari, Gerber, M. S., Barnes, L. E. (2017). Hdltext: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pp. 364–371.
- Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Loquercio, A., Maqueda, A. I., Maqueda, A. I., del-Blanco, C. R., & Scaramuzza, D. (2018). Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2), 1088–1095.
- Madzarov, G., Gjorgjevikj, D., & Chorbev, I. (2009). A multi-class svm classifier utilizing binary decision tree. *Informatica*, 33(2), 233–241.
- Perico, D.H., Silva, I.J., Junior, C.O.V., Homem, T.P.D., Destro, R.C., Tonidandel, F., Bianchi, R.A.C. (2014). Newton: a high level control humanoid robot for the robocup soccer kidsize league. In *Robotics*, pp. 53–73. Springer.
- Perico, D.H., Silva, I.J., Vilao, C.O., Homem, T.P.D., Destro, R.C., Tonidandel, F., Bianchi, R.A.C. (2014). Hardware and software aspects of the design and assembly of a new humanoid robot for robocup soccer. In *Robotics: SBR-LARS robotics symposium and robocontrol (SBR LARS Robocontrol), 2014 joint conference on*, pp. 73–78, Oct.
- Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3), 221–234.
- Rhinehart, N., McAllister, R., & Levine, S. (2019). Deep imitative models for flexible inference, planning, and control. arXiv preprint arXiv:1810.06544.
- RoboCup Federation. Robocup soccer humanoid league laws of the game. <https://humanoid.robocup.org/materials/rules/>, (2020).
- Roy, D., Panda, P., & Roy, K. (2020). A hierarchical deep convolutional neural network for incremental learning: Tree-cnn. *Neural Networks*, 121, 148–160.
- Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: a modern approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.
- Schnekenburger, F., Scharffenberg, M., Wülker, M., Hochberg, U., Dorer, K. (2017). Detection and localization of features on a soccer field with feedforward fully convolutional neural networks (fcnn) for the adultsized humanoid robot sweaty. In *Proceedings of the 12th workshop on humanoid soccer robots, IEEE-RAS international conference on humanoid robots, Birmingham*.
- Speck, D., Barros, P., Weber, C., Wermter, S. (2016). Ball localization for robocup soccer using convolutional neural networks. In *Proceedings of the 2016 RoboCup international symposium*, pp. 19–30. Springer.
- Speck, D., Bestmann, M., & Barros, P. (2018). *Towards real-time ball localization using cnns, Robot World Cup XXII*. Berlin: Springer.
- Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105, 2295–2329.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.
- Team description paper (2017). Citbrains team (kid size league).
- Ting, K.M. (2010). *Precision and Recall*, pp. 781–781. Springer US, Boston, MA.
- van Dijk, S.G., Scheunemann, M.M. (2018). Deep learning for semantic segmentation on minimal hardware. *arXiv preprint arXiv:1807.05597*.
- Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., Yu, Y. (2015). Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. *2015 IEEE international conference on computer vision (ICCV)*.
- Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3), 55–75.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.