





Real-Time Scene Understanding Using Deep Neural Networks for RoboCup SPL

Marton Szemenyei¹  and Vladimir Estivill-Castro² 

¹ Budapest University of Technology and Economics, Budapest, Hungary
szemenyei@iit.bme.hu

² Griffith University, Brisbane, QLD, Australia

Abstract. Convolutional neural networks (CNNs) are the state-of-the-art method for most computer vision tasks. But, the deployment of CNNs on mobile or embedded platforms is challenging because of CNNs' excessive computational requirements. We present an end-to-end neural network solution to scene understanding for robot soccer. We compose two key neural networks: one to perform semantic segmentation on an image, and another to propagate class labels between consecutive frames. We trained our networks on synthetic datasets and fine-tuned them on a set consisting of real images from a Nao robot. Furthermore, we investigate and evaluate several practical methods for increasing the efficiency and performance of our networks. Finally, we present RoboDNN, a C++ neural network library designed for fast inference on the Nao robots.

Keywords: Computer vision · Deep learning · Semantic segmentation · Neural networks

1 Introduction

Deep learning [10] is rapidly revolutionising the field of computer science. While deep neural networks (DNNs) have many usages, they are now undoubtedly the most popular technique in the field of intelligent perception, especially computer vision. In RoboCup, especially in the SPL league, several teams [7, 12, 15, 17, 19] have used CNNs to classify relevant objects on the soccer field. However, due to the limitations of the robot's hardware, these networks were relatively shallow and were designed to classify fixed-resolution image patches only. This straightforward design meant that the teams had to use separate object-proposal methods to feed their network with candidate image regions.

We propose an end-to-end real-time object detection method for the Nao robots using deep neural networks. Our method combines two separate networks to achieve high accuracy at reasonable speed. The first network is a deep neural network trained to perform semantic segmentation (pixel-wise classification) on the image from the robot's camera. The second is a smaller network, trained to propagate the class labels from the previous image onto the next. Our system

is capable of localizing four foreground classes (ball, robot, goalpost, and field line), without the use of a separate object proposal system.

Furthermore, we also report on the comparison of methods to accelerate deep neural networks. These techniques include using field edge detection to reduce the image size without decreasing the resolution of the relevant image parts, and pruning the weights of the neural network. We also compare with different network structures and show how to find the optimal accuracy/runtime trade-off.

Finally, we present RoboDNN, a lightweight C++ deep neural network library. RoboDNN is a forward-only library, designed for maximum performance on the Nao robot. The library has no dependencies, and does not use new constructs of the C++11 standard, facilitating compilation for the Nao. Moreover, we designed RoboDNN using the strictest compiler settings. Our implementation also offers compatibility with the popular Pytorch [1] DNN framework, with the ability to import neural nets trained using Pytorch. The RoboDNN library, the code and datasets are all available on our website [2].

2 Related Work

The availability of high-quality datasets and computational resources has enabled the training DNNs, once some numerical problems were surpassed [9]. The use of deep learning is perhaps most prominent in the field of computer vision, where DNNs are used for standard classification [13] and (semantic or instance) segmentation tasks [21, 23]. Fast architectures, such as YOLO [20] exist for object detection, however, segmentation provides a more complete understanding of the scene. Also YOLO struggles with cluttered objects, which is frequent in robot soccer scenes.

Semantic segmentation aims to achieve visual scene understanding. The objective is to segment the image by classifying each pixel individually. The simplest way to achieve this with neural networks is to use a classification network, and replace the final fully connected layers by (usually 1×1) convolutional layers. This network can output a segmented image at a lower resolution, which can be upsampled using techniques, such as bilinear upsampling [5]. Other works [25] employ a superpixel segmentation method, and classify these superpixels individually in order to approximate object boundaries with higher accuracy.

The SegNet architecture [5] uses so-called unpooling layers to upsample the feature maps. In SegNet the max pooling layers in the first half of the network store the index of the maximum value and share this information with the corresponding unpooling layer. This extra piece of information allows the unpooling layer to recover the spatial information lost during downscaling. Nonetheless, the full feature map is not recovered, since the non-maximum values are permanently lost. Several important advances have improved the accuracy of SegNet-based segmentation, especially when it comes to capturing the fine details of objects. The first of these improvements is the Fully Convolutional Network (FCN) architecture [23], which introduced shortcuts (skip connections) from the front layers of the network. By adding shortcuts from early layers, the final layer has more

information on the fine-resolution details of the image, resulting in better approximation of object boundaries. Shortcuts also improve the convergence properties of deep neural networks considerably [11].

Choosing the upscaling and downscaling methods in the network can also affect the performance significantly. The FCN network uses strided convolutions instead of pooling. It introduces costly transposed convolutional layers instead of unpooling to implement learnable downsampling and upsampling operations [23]. Wang et al. [24] introduced the dense upsampling convolution operation (DUC), which was shown to increase the accuracy further, but again, at a considerable increase of computational cost.

The field of view of the final classification neuron also influences the accuracy of the segmentation network, since it determines the amount of contextual information the final neuron can use to determine the class of each pixel. Chen et al. [6] showed that using dilated/atrous convolutional filters increases the performance without increasing the computational cost. Pooling operations may also be atrous [6], resulting in a similar improvement.

While convolutional neural networks have achieved staggering accuracy in numerous applications, their power comes with a high computational cost. Such high computational cost seems to prohibit the use of CNNs on board of mobile and embedded platforms. While several methods have been proposed for reducing the size and computation required for neural networks, the most relevant to our application is pruning [4]. During the pruning process, a ranking method is used to order the weights or neurons of a layer by importance, then a fixed percentage of the least important neurons/connections are deleted/set to zero. Next, the network is fine-tuned while keeping the pruned elements at zero value. Several ranking methods exist ranging from brute-force methods, that only use the magnitude of the weights to more complex ones, such as pruning weights so that the change in the network's loss function would be minimal [18].

Computer Vision in RoboCup. Achieving human-level vision and scene understanding is an essential component of achieving RoboCup's goals and the RoboCup environment has steadily changed from featuring objects that are easy to recognize using low-level features, such as colour, to ones that greatly resemble objects used in human soccer. The vision pipelines used by the competing teams have changed in tandem, going from human-engineered vision methods [16, 22] to pipelines relying increasingly on machine learning. Several teams have used convolutional neural networks either for binary classification tasks [7, 17] or to detect several relevant object categories [15, 19]. These methods, however, use CNNs for classification only, therefore they still require a separate object proposal method, and the quality of the system may largely depend on the efficiency of the algorithm used to generate candidates for classification. A further disadvantage is that running the same neural network on potentially overlapping image regions is wasteful, since the same features are computed twice.

Hess et al. [12] present a high-quality virtual RoboCup environment created in Unreal Engine. Their work allows to easily create large datasets of realistic

images of a soccer field along with pixel-level semantic labelling. Since the performance of a trained neural network is highly dependent on the quality and quantity of the training data, and creating a large hand-labelled database is highly time-consuming, their work was profoundly valuable for our research.

3 Preparation of the Training Data

To ensure the quality and amount of the training data for training the semantic segmentation network, we created a synthetic image set of 5000 images [12]. We used 100 different random sets of environmental variables, and generated 50 images with each setting. The images were separated into train and test sets randomly, using an 80–20 division. The automatically generated labels are available as PNG images, and contain labels for all five relevant categories (background, ball, robot, goal and line).

In addition, we extended the challenge to allow for the creation of image sequences instead of independent scenes. We used this mode to create a dataset of 800 images to train the label propagation network. This database also features 100 individual image sequences with different random scene parameters. In each sequence, however, the position and orientation of the camera and field objects only changes marginally between consecutive frames.

Synthetic images are an excellent way of pre-training a network on a large dataset, yet due to the differences between a synthetic and a real environment we require a database of real images to fine-tune the network. But the pre-training allows a much smaller dataset for fine tuning than would be otherwise required. For these reasons, we created a real semantic segmentation database consisting of 570 images taken at 3 separate locations: at the venue of RoboCup17, at the venue of IJCAI17 and in our lab at Griffith University. The images are from the top camera, since these usually contain more complex scenes, justifying the use of neural networks. A portion of this database consists of image sequences, which are used as a dataset for label propagation.

We manually annotated the images using a tool of our own creation. Our tool provides several ways to aid the annotation process, such as tools for drawing polygons and lines, as well as square and circular brush tools. The program also uses the superpixel segmentation method proposed by Li and Chen [14] to speed up the labelling process. In the case of successive images, the tool is able to use dense optical flow to approximate the labels of the next image. Using the tool, it is also possible to mark the edges of the field, setting pixels and labels outside the field to black and background respectively. This dataset can be used for detection easily by computing bounding boxes for the connected label components.

Despite having a fair number of real images, they were considerably less varied than the synthetic images, since they included only three locations with their unique environmental settings (such as lighting and carpet colour). To compensate for this disadvantage, we used aggressive, unique data augmentation in addition to standard techniques, such as flipping images horizontally. To emulate changes in lightning conditions, we applied random changes in the

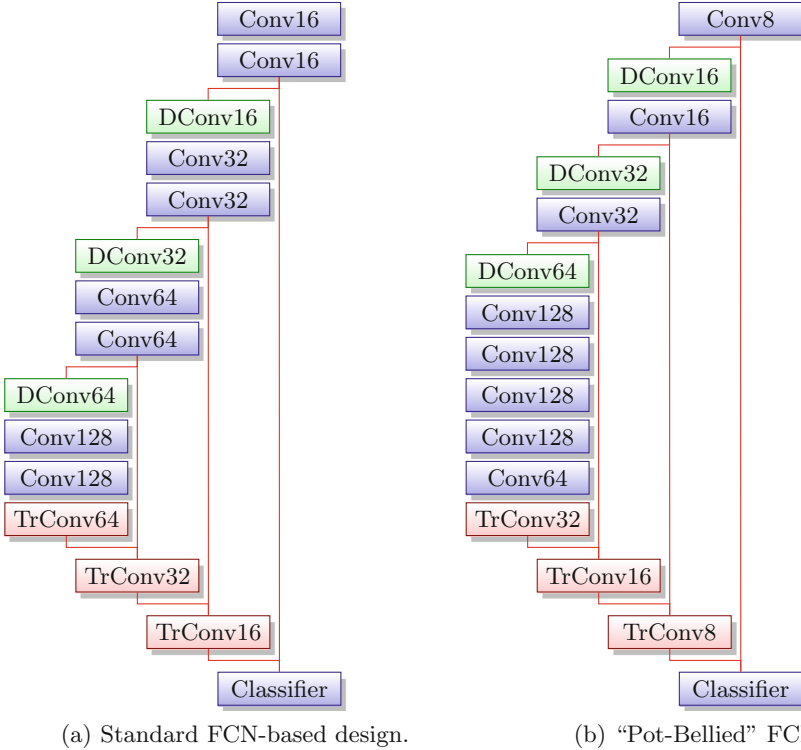


Fig. 1. Our architectures: green nodes are strided convolution, while red nodes are transposed convolution. We have batch normalization in every node. (Color figure online)

brightness and contrast of the images. To introduce further variation into the dataset, we also applied random shifts to the hue and saturation of our pictures, which may help the robots with unique carpet colours. In Sect. 6 we show that our data augmentation techniques improve the accuracy of the trained models greatly.

4 Model Selection and Training

We used the Pytorch framework [1] for training the network. We applied stochastic gradient descent (SGD) optimisation with momentum and weight decay regularisation. During training, we used an adaptive learning rate schedule, which reduced the learning rate of the network after N consecutive epochs in which the validation loss could not fall below the current lowest value. We made a slight modification to Pytorch’s learning rate scheduler, to allow us to reload the current best model in every learning-rate reduction event. We remark that this

variation results in the optimiser finding a new optimum more often after reducing the learning rate. Table 1 displays the hyperparameters used for pre-training and fine-tuning the semantic segmentation and label propagation networks.

The nature of the scenes specific to robot soccer fields offers a major challenge. Typical soccer images offer few pixels belonging to objects of interest. In our datasets, the ratio of background pixels is around 93–94%. Moreover, the rest distribute somewhat unevenly amongst the relevant classes. This uneven dataset usually complicates convergence and may result in a final network that is heavily biased towards making false negative-type errors. We stress that this imbalance stems from the distribution of classes *in the individual images themselves*. Therefore it is not possible to re-sample the training set.

We propose two solutions to overcome this difficulty. First, we selected the images for the training set so that they would contain a relatively high percentage of pixels from objects of interest (so called foreground). Second, we used a weighted version of the 2-dimensional negative log-likelihood (NLL) loss function, which is implemented in Pytorch, encouraging the network to emphasize more the relevant object categories.

The next challenge is to define an efficient and powerful network structure. Our review of the literature suggests an architecture based on FCN [23], using strided convolution for downsampling and transposed convolution for upsampling, as well as employing dilated convolutions to increase the field of view of the final classification layer. But, we avoided using DUC for upsampling, due its higher computational requirements. This first design (refer to Fig. 1a) had three modules consisting of convolutional and downsampling layers, combined with three upsampling layers.

Most CNNs used for semantic segmentation are relatively waist-heavy, meaning that the middle section of network, where the feature map has the smallest spatial dimensions has the largest number of filters. This has obvious advantages when it comes to memory consumption and computational efficiency. In our experiments, we decided to push this feature even further, using few and shallow layers to downsample the feature map quickly, then using a larger number of deep convolutional layers at the lowest level, followed by a similarly shallow and quick upsampling. In Sect. 6 we demonstrate that this network structure is much more efficient, providing better accuracy for lower computational cost. Figure 1b illustrates our new alternative, the “Pot-Bellied” (PB-FCN) architecture.

Our training procedure consists of three steps. First, the first half of the segmentation network is trained on the dataset provided using Hess et. al [12] generation for classification. We modified the dataset by separating the background and field line classes. Next, the full segmentation network is trained on the synthetic database. Finally, the full segmentation network is fine-tuned on the real database. The training procedure for the label propagation network is similar, except the first step is omitted.

Table 1. Hyperparameters used for the training procedures.

Parameter	LR	Momentum	Decay	Reduction	Patience	Batch	Epochs
Segmentation	0.1	0.5	1e-3	0.5	20	32	200
Fine-tune	0.01	0.1	1e-3	0.5	50	8	500
Label prop	0.2	0.5	1e-5	0.5	10	16	100
LP fine-tune	0.05	0.1	1e-5	0.5	25	8	250

5 Real-Time Implementation

To ensure real-time performance of our object detection pipeline, we must employ several techniques to improve the speed of the trained neural network. For the Nao hardware, these improvements are critical, since the networks resulting from training as in the previous section require approximately 1 s to run on a Nao V5 robots using the Darknet library [20]. For all our experiments, we used 160×120 images in YUV colour space.

The first technique we employ is weight pruning: since convolution is implemented as a matrix multiplication, setting weights to zero increases the efficiency significantly, even when an extra operation (checking if the weights are zero) is introduced. We used a brute-force pruning technique, simply setting 75% of the smallest weights in every layer to zero, and then fine-tuning the network, while forcing the pruned weights to remain zero. We found, that this technique reduced the runtime of the network by approximately 70%.

Moreover, our vision pipeline includes a hand-crafted field detection system, which is used by our network to crop part of the image (the outside the field is usually the top part of the image). This approach comes with two advantages. First, it reduces the number of pixels to be processed without reducing the level of detail. Second, if the network is trained on images where the parts outside the field are omitted, it avoids learning complex backgrounds outside the field (which are easily confused with field objects). While this technique provides considerable improvement in the networks speed, this improvement is highly dependent on the robot’s position in the field. For this reason, we used uncropped images when comparing the execution times of different models and methods.

Our second technique for increasing the speed of our pipeline is label propagation. Here, we estimate the labels of the next image by using the labels of the previous one. We can achieve a considerable increase in speed, by only running the main neural network every 10 or 20 frames, and provided that accurate label propagation can be implemented using a significantly faster algorithm.

We first employed Gunnar Farneback’s dense optical flow (OptFlow method) algorithm [8] to move the labels to their new location. While the algorithm’s speed was satisfactory, the accuracy suffered. Namely, small, single-pixel errors would accumulate over time, constantly eroding small objects. Also, the optical flow-based method is completely unable to handle faster movements or new objects appearing in the image (or partially seen objects sliding in).

Table 2. Accuracy variations from the *baseline* by the training techniques.

Technique	Baseline	Augment.	Field	Reload	Prune
TPA	97.72	0.61	0.31	0.09	-0.18
MCA	92.19	2.55	0.82	1.08	-0.8
MIoU	75.57	6.66	2.36	0.19	-0.4

This problem can be remedied by training a neural network to predict the labels of the next image from the labels of the previous one. Since this is much easier, than predicting the labels from the raw image, we used considerably smaller version of PB-FCN (Fig. 1), which would run at approximately twice the speed of the segmentation network (PB-FCN-LP method). The network takes an 8-channel input, consisting of the Y channels of the two images, their difference and the 5-channel label image. For numerical reasons, the binary labels were scaled between -1 and 1 . The label propagation network was trained using sequential the synthetic and real datasets mentioned in Sect. 3. We used the same data augmentation techniques, and trained the network to be able to predict the new labels in both ways (previous-to-next and vice versa), since the vast majority of movements might occur in both directions.

Since this method combines knowledge about the visual appearance of the classes and the movement between the images it is arguably able to account for the appearance of new objects and handle larger movements. Moreover, for the same reason, the label propagation network has some form of self-correcting ability, thus misclassifying a pixel in one frame does not mean that the error will be carried on until the next run of the segmentation network. We observed during our experiments that the label propagation network seemed to be able to incrementally correct the mistakes made by the segmentation network, especially when the robot was not moving (Fig. 2).

While implementation could use an existing library, the target platform (Nao robot) is a challenge. For example, Caffe [3] is a relatively old library with numerous dependencies, making it difficult to compile for the Nao robot. While the newer Darknet [20] has no dependencies, it lacks support for several important features we used in our design, such as dilated convolutions and affine batch normalization. Thus, we created our own C++ library called RoboDNN, based on Darknet, implementing the most common neural network layers. Our library is designed for inference only, therefore all code for training the networks was stripped. Our library has no external dependencies, does not require C++11, and - like all of MiPal's code - compiles using the strictest compiler settings.

The current version of RoboDNN is compatible with Pytorch. Our code includes support for dilated convolutions, output padding for transposed convolutions, and layers for affine batch normalization. Thus, RoboDNN is fully compatible with neural networks trained in Pytorch, and we provide code to export the weights Pytorch models along with the library. Our library is also

Table 3. Comparison of the different neural network architectures.

Model	FCN	PB-FCN	PB-FCN-VGA	ResNet-DUC	OptFlow	PB-FCN-LP
TPA	98.42	98.50	98.87	98.71	95.82	96.52
MCA	94.95	94.40	96.50	94.88	86.15	90.7
MIoU	80.31	81.30	84.00	83.98	82.70	79.15

optimized for maximum efficiency, including support for accelerating pruned networks, running on cropped images and several in-place operations for memory efficiency.

6 Results

We now demonstrate experimentally the virtues of our design and training method. First we evaluate the accuracy across model design and learning techniques. Second, we assess the speed of our pipeline on the Nao V5 robots. Figure 3 shows some of the best and worst results of the segmentation.

Tests on Accuracy. We use three measures to evaluate accuracy. The first is the percentage of pixels classified correctly, called Total Pixel Accuracy (TPA). Taking the average of TPA per class is what we call Mean Class Accuracy (MCA). These two measures are relevant because of our class imbalance. Note that TPA will favour models that are more likely to err on the side of background, but MCA will be higher for models that are more likely to make false positive predictions. The third measure is Mean Intersection over Union (MIoU), which (on our dataset) prefers models with minimal confusion between foreground classes.

We now demonstrate the change in accuracy provided by our data augmentation operations, field extraction, reloading learning rate scheduler, and pruning. We measured the improvements of these techniques separately on a fine-tuned PB-FCN network. Table 2 presents variations in accuracy from the baseline. The results indicate that our techniques increase the accuracy considerably, while our brute-force pruning retains most of the predictive power.

Table 3 compares four different models: The standard FCN-based model, our PB-FCN used on both 160×120 and 640×480 (PB-FCN-VGA) resolution images, and a fourth model using ResNet152 [11] and deep DUC upsampling. From these results, we can draw several conclusions. First, PB-FCN is slightly more powerful compared to the standard FCN structure. Second, a comparatively shallow PB-FCN loses surprisingly little accuracy compared to the ResNet152 model, and even outperforms it considerably when used at VGA resolution.

Lastly, we compare our pruned, fine-tuned PB-FCN-based label propagation network with the optical-flow based method (see Table 3). Note that neural label propagation clearly outperforms the optical-flow method. Moreover, optical flow

Table 4. Detection results on our validation set.

Model	PB-FCN	PB-FCN-VGA	ResNet-DUC	BBN-L	BBN-MC	BBN-L	BBN-MC
Accuracy	95.53	98.86	95.36	83.36	81.39	95.39	94.89
False pos.	2.69	1.00	2.32	26.19	21.58	5.27	4.45
False neg.	4.47	1.14	4.64	16.64	18.61	4.61	5.11

Table 5. Comparison of the execution times of different models.

Model	PB-FCN	Pruned	VGA	FCN	ResNet	Opt-flow	Neural LP	BBN-L	BBN-MC
Time (ms)	1480	380	2850	640	8000	80	170	70	22
FpS	0.7	2.6	0.35	1.56	0.125	12.5	5.9	14	46

produces slightly better MIOU results due to the heavily unbalanced dataset (optical flow introduces minimal confusion between foreground classes). We used the real-image dataset for all the results in this section.

Comparison Against Other Solutions. We also compare our results against the networks used by other teams. First, we evaluate the first half of our PB-FCN network against the classification results reported by Hess et al. [12] using the same test dataset. Our algorithm clearly outperforms theirs, producing 96.66% accuracy compared to 94.4% and 93.52% with their BBN-L and BBN-M-C models respectively. Moreover, we managed to achieve this result on a significantly smaller dataset, consisting of 9,000 images per class only (compared to 25,000 per class). This improvement is largely due to our data augmentation methods.

Since to our knowledge this is the first work to publish semantic segmentation results in the context of robot soccer, there is no established baseline to compare our method against. We are also aware that other teams may use an object detection method instead of a pixel-wise classification algorithm. For this reason, Table 4 presents the performance of our networks in a detection task by simply counting the percentage of relevant objects that were correctly detected. Note, that a portion of false negatives and false positives may be objects that were incorrectly merged or separated by our network.

We also evaluated BBN-L and BBN-M-C on this dataset by providing it with all the relevant image patches, as well as three background patches per image to measure the false positive rate. This setting is equivalent to an object candidate generation method with zero false negative rate and here our PB-FCN clearly outperforms the two other networks. The comparison is fair since we test all classification networks on patches extracted from our training database.

Evaluating Execution Time. We tested the execution time of our entire vision pipeline on a Nao V5 robot, using the top camera image. We used a single core to run the neural network, and we ran the pipeline with other soccer subsystems active. Table 5 shows the execution time of the pruned versions of

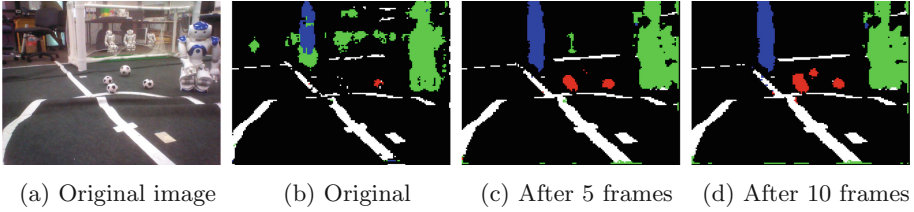


Fig. 2. The self-correcting ability of the label propagation network.

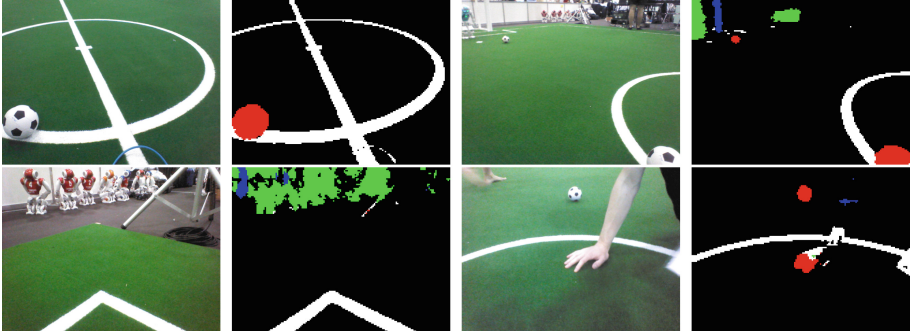


Fig. 3. A few examples of good (top) and bad (bottom) results.

the models compared in the previous subsection. For reference, we also included the non-pruned version of PB-FCN. The results show a clear improvement as a result of pruning, and that PB-FCN outperforms the vanilla FCN in speed as well. We remark that the data shows that running a relatively shallow network on a higher resolution image seems to be much faster than running ResNet on a downscaled version, while providing superior accuracy.

In Table 5 we also present the comparison of label propagation using optical flow and CNNs. The results show that the extra accuracy coming with the neural network comes at lower speeds. Still, the fully neural vision pipeline runs at 6 frames per second, which is sufficient to enable real-time reactions at robot soccer speeds. For reference, we include our measurements of the speed of the neural network used by Hess et al. [12]. The comparison shows, that although we could achieve significant improvements in accuracy and the neural network’s efficiency, it is still several times slower than other methods.

7 Conclusion

In this paper, we presented a deep neural network-based method for scene understanding in the context of robot soccer. Our method uses a semantic segmentation network and a separate label propagation net to increase the frame rate of the vision system. With our experiments, we demonstrated the efficiency of our method, including the improvements we achieved using our data augmentation

techniques, pruning and field-edge cropping. Our method has superb accuracy at satisfactory speed.

We also presented large semantic segmentation and label propagation datasets consisting of synthetic images, as well as small real datasets for the same tasks, including a tool for manual pixel-wise labelling of images. Finally, we presented a Pytorch-compatible C++ deep neural network library designed for fast inference on the Nao robots supporting the acceleration techniques discussed in this paper. Our library has been designed to compile for the Nao robots using the strictest compiler settings.

Acknowledgments. Our research was supported by NVIDIA and Erasmus Mundus PANTHER.

References

1. <http://www.pytorch.org>
2. <http://3dmr.iit.bme.hu/research/robocup/index.html>
3. <http://caffe.berkeleyvision.org>
4. Anwar, S., Hwang, K., Sung, W.: Structured pruning of deep convolutional neural networks. *ACM J. Emerg. Technol. Comput. Syst.* **13**(3), 1–11 (2017)
5. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(12), 2481–2495 (2017)
6. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **PP**(99), 1 (2017)
7. 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. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) *RoboCup 2017*. LNCS, vol. 11175, pp. 19–30. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_2
8. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) *SCIA 2003*. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45103-X_50
9. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256 (2010)
10. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (2016)
12. Hess, T., Mundt, M., Weis, T., Ramesh, V.: Large-scale stochastic scene generation and semantic annotation for deep convolutional neural network training in the RoboCup SPL. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) *RoboCup 2017*. LNCS, vol. 11175, pp. 33–44. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_3
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998)

14. Li, Z., Chen, J.: Superpixel segmentation using linear spectral clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1356–1363 (2015)
15. Menashe, J., et al.: Fast and precise black and white ball detection for RoboCup soccer. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) RoboCup 2017. LNCS, vol. 11175, pp. 45–58. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_4
16. Metzler, S., Nieuwenhuisen, M., Behnke, S.: Learning visual obstacle detection using color histogram features. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 149–161. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32060-6_13
17. Javadi, M., Azar, S.M., Azami, S., Ghidary, S.S., Sadeghnejad, S., Baltés, J.: Humanoid robot detection using deep learning: a speed-accuracy tradeoff. In: Akiyama, H., Obst, O., Sammut, C., Tonidandel, F. (eds.) RoboCup 2017. LNCS, vol. 11175, pp. 338–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_28
18. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. [arXiv:1611.06440](https://arxiv.org/abs/1611.06440) (2016)
19. 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, vol. 11175, pp. 398–409. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00308-1_33
20. Redmon, J.: Darknet: open source neural networks in C (2013–2016). <http://pjreddie.com/darknet/>
21. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
22. Schwarz, I., Hofmann, M., Urbann, O., Tasse, S.: A robust and calibration-free vision system for humanoid soccer robots. In: Almeida, L., Ji, J., Steinbauer, G., Luke, S. (eds.) RoboCup 2015. LNCS, vol. 9513, pp. 239–250. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-29339-4_20
23. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(4), 640–651 (2017)
24. Wang, P., et al.: Understanding convolution for semantic segmentation. [arXiv:1702.08502](https://arxiv.org/abs/1702.08502) (2017)
25. Xing, F.Z., Cambria, E., Huang, W.B., Xu, Y.: Weakly supervised semantic segmentation with superpixel embedding. In: IEEE International Conference on Image Processing (ICIP), pp. 269–273 (2016)