



# Large-Scale Stochastic Scene Generation and Semantic Annotation for Deep Convolutional Neural Network Training in the RoboCup SPL

Timm Hess<sup>1</sup>(✉), Martin Mundt<sup>1,2</sup>, Tobias Weis<sup>1</sup>, and Visvanathan Ramesh<sup>1,2</sup>

<sup>1</sup> Software Engineering for Computer Vision, Bembelbots, Goethe University Frankfurt, Frankfurt am Main, Germany

{hess,weis}@ccc.cs.uni-frankfurt.de

<sup>2</sup> Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main, Germany

{mundt,ramesh}@fias.uni-frankfurt.de

**Abstract.** Object detection and classification are essential tasks for any robotics scenario, where data-driven approaches, specifically deep learning techniques, have been widely adopted in recent years. However, in the context of the RoboCup standard platform league these methods have not yet gained comparable popularity in large part due to the lack of (publicly) available large enough data sets that involve a tedious gathering and error-prone manual annotation process. We propose a framework for stochastic scene generation, rendering and automatic creation of semantically annotated ground truth masks. Used as training data in conjunction with deep convolutional neural networks we demonstrate compelling classification accuracy on real-world data in a multi-class setting. An evaluation on multiple neural network architectures with varying depth and representational capacity, corresponding run-times on current NAO-H25 hardware, and required sampled training data is provided.

## 1 Introduction

By the middle of the 21st century artificial intelligence based humanoid soccer robots are envisioned to win against a human team in a football game complying with official FIFA rules. At all times an instance model of the current situation is required to enable the complex interplay of sensing, control and prediction. Detection and classification of the agent's surroundings are essential constituents in the visual component of such a world model. By successively alleviating environmental constraints, e.g. illumination and color cues, the RoboCup challenge strives to capture real-world complexity. This results in greater need for efficient, robust real-time computer vision systems. The contemporary landscape of RoboCup research is at large dominated by model-driven approaches relying heavily on human engineered vision pipelines [1–4] that require substantial

---

T. Hess and M. Mundt have contributed equally.

© Springer Nature Switzerland AG 2018

H. Akiyama et al. (Eds.): RoboCup 2017, LNAI 11175, pp. 33–44, 2018.

[https://doi.org/10.1007/978-3-030-00308-1\\_3](https://doi.org/10.1007/978-3-030-00308-1_3)

amounts of domain expertise to construct. Due to the rise of computational power and availability of large data sets deep convolutional neural networks (CNNs) have increased in popularity in both academia and industry and have been shown to perform exceptionally well on many vision tasks in the course of the last years [5–7]. With [8,9], only recently some of these advances have been applied to the Standard Platform League (SPL). Both of these works focus on binary classification of (NAO) robots and balls alone.

However, the successful training of deep neural networks depends on extensive, tediously gathered, curated and annotated data sets. To generalize well these data sets need to span the space of potential inputs as thoroughly as possible, oftentimes rendering deep neural network approaches infeasible. Building on advances in computer graphics and respective publicly available graphics-/game-engines (e.g. [10,11]) this issue can in principle be addressed by resorting to generated synthetic instances of the domain in question. Practicality in several domains has been demonstrated in the context of deep learning [12–16], although a careful consideration of the underlying assumptions in the generative model with respect to the task is crucial to avoid statistical mismatches in the data distributions that ultimately determine the overall viability of data driven approaches.

In this work we develop an automated framework, using a state of the art real-time rendering engine [11], for the generation of semantically annotated images of simulated SPL scenes through systematic mapping of geometric and photometric priors derived from specifications (e.g. [17]). In a series of experiments we demonstrate compelling results, evaluate tradeoffs in choice of different CNN architectures with respect to accuracy and runtime performance on the NAO-H25 robot and present insights in terms of required sampling density. Due to the modular nature of the simulation framework it is readily extensible to novel conditions coinciding with the goal of the RoboCup. To promote transparency and reproducibility in research, we open-source our contributions at <https://github.com/TimmHess/UERoboCup>.

## 2 Generative Scene Model

First, objects and their relative pose expressed in the form of meshes are assigned to geometrical parameters, whereas properties related to lighting and respectively the scattering, transmission and reflection thereof fall into the category of photometric parameters. We formalize the priors known from specifications in form of distributions and formulate the scene generation process in terms of stochastic sampling.

### 2.1 Geometric Parameters

A typical scene in a SPL match is comprised of a limited set of objects, that is one ball, a maximum of ten robots, two goals and the playing field, as well as a set of light sources. Being entirely static in nature, the playing field  $\mathbf{F}$ , with a

spatial extent of  $\mathbf{F}_w \times \mathbf{F}_h = 9\text{ m} \times 6\text{ m}$  [17], defines the geometrical boundaries for the placement of the other objects in a two-dimensional Cartesian coordinate system. To ensure approximately equal numbers of objects in the camera field of view for the later sampling of photometric parameters, the ball  $\mathbf{B}$  is chosen as the central component in the stochastic scene generation process. We define the distribution on its spatial position to stem from two independent uniform distributions:

$$p(\mathbf{B}_{x,y}) = (\mathcal{U}(0, \mathbf{F}_w), \mathcal{U}(0, \mathbf{F}_h)). \quad (1)$$

A first robot  $\mathbf{R}_{x,y,\alpha}^{\text{cam}}$ , parametrized by its spatial position  $x, y$  and angle  $\alpha$  from whose viewpoint the scene will later be rendered, is sampled such that it has a distance  $d$  to the ball and is placed randomly on the circle defined by above radius:

$$p(\mathbf{R}_{x,y}^{\text{cam}}|\mathbf{B}_{x,y}) = \mathbf{B}_{x,y} + \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} [0\ d]^\top. \quad (2)$$

Here distance  $d \sim \mathcal{U}(0.3\text{ m}, 1.5\text{ m})$ , matched to the robots' static head pose with lowest pitch (spanning the vertical field of view), and angle  $\phi \sim \mathcal{U}(0, 2\pi)$  are also chosen to be uniformly distributed. In order to vary the horizontal position of the ball in the camera's field of view, an angular offset  $\gamma \sim \mathcal{U}(-30.5, +30.5)\pi(180)^{-1}$  is uniformly sampled corresponding to the horizontal field of view of current NAO hardware [18]. Letting  $\beta$  be the angle between  $\mathbf{R}_{x,y}^{\text{cam}}$  and  $\mathbf{B}_{x,y}$  in the field's coordinate system, then the angle of the robot is given by

$$p(\mathbf{R}_\alpha^{\text{cam}}|\beta) = \beta + \gamma. \quad (3)$$

$N_R$  (in a typical game  $N_R = 9$ ) other robots are uniformly placed on the field according to

$$p(\mathbf{R}_{x,y,\alpha}^i) = (\mathcal{U}(0, \mathbf{F}_w), \mathcal{U}(0, \mathbf{F}_h), \mathcal{U}(0, 2\pi)), \quad i \in \{0, \dots, N_R\}. \quad (4)$$

Three different types of robots are considered (standing, sitting and lying), which we model using a categorical distribution with probabilities 0.8, 0.1 and 0.1, that have not been denoted in equations for the sake of simplicity.

Currently light sources are placed in an evenly spaced  $3 \times 2$  grid 3 m above the field. The arrangement and quantity can in principle also be sampled from any distribution but has not yet been included in the current model.

## 2.2 Photometric Parameters

Consistent with a physics based model, light sources  $\mathbf{L}$  are characterized through intensity  $\mathbf{L}_I$  and temperature  $\mathbf{L}_T$  [19], the latter effectively defining the illuminants' color. Given a set of  $N_I$  intensities, each represented by a normal distribution, we sample a light intensity according to

$$p(\mathbf{L}_I|k) = \mathcal{N}(\mu_k, \sigma_k), \quad k \sim \mathcal{U}(1, N_I), \quad (5)$$

assuming a range from 1700 lm, corresponding to a 100 W light-bulb [20], to 4000 lm approximating an upper limit of current consumer LED flood lights. The

intensity value  $\mathbf{L}_I$  in Eq. 5 is applied to all light sources in the scene, expressing the belief that venues are constructed in a self-consistent manner. The variance parameter  $\sigma_k$  models small perturbations as consequence of wear, current fluctuations and other forms of minor but non-negligible deviations and is thus assumed to be a small constant.

We restrict illuminant colors to follow the black body locus with  $D_{65}$  standard illuminant between temperatures of  $T_{\text{low}}$  and  $T_{\text{high}}$ , and sample uniformly from this space:

$$p(\mathbf{L}_T) = \mathcal{U}(T_{\text{low}}, T_{\text{high}}). \quad (6)$$

Reasonable temperatures range from 3000 K to 12000 K, spanning light-colors from yellow to white to blue.

As noted in the specifications of the SPL league, the playing field is restricted to be of green color. For ease of notation, we use the HSV colorspace and sample uniformly from

$$p(\mathbf{F}_{H,S,V}) = (\mathcal{U}(H_{\text{low}}, H_{\text{high}}), \mathcal{U}(S_{\text{low}}, S_{\text{high}}), \mathcal{U}(V_{\text{low}}, V_{\text{high}})), \quad (7)$$

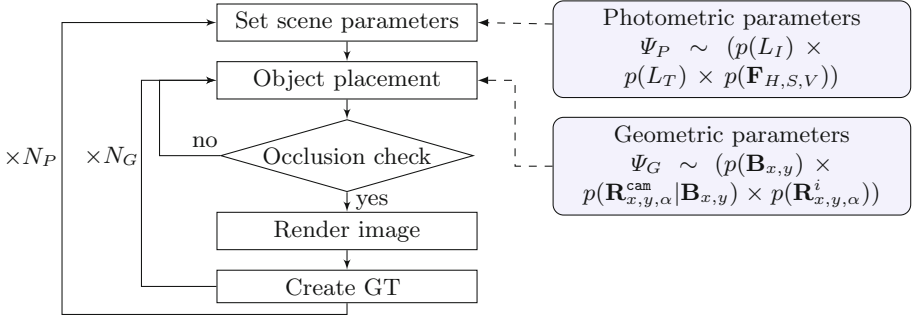
where the hue is set to be between  $H_{\text{low}} = 100^\circ$  and  $H_{\text{high}} = 140^\circ$  resembling shades of green.  $S_{\text{low}} = 0.5$ ,  $S_{\text{high}} = 1.0$  and  $V_{\text{low}} = 0.25$ ,  $V_{\text{high}} = 1.0$  determine saturation and brightness. Lower limits of  $S$  and  $V$  have been chosen to exclude under-saturated and too dark colors.

### 3 Rendering and Semantic Annotation Workflow

Even though sophisticated ray-tracing rendering engines are capable of producing highly photo-realistic images [10], the use of intricate sampling techniques usually comes at the expense of high computational complexity. At the same time, modern deep learning methods require tremendous amounts of data to achieve state-of-the-art results.

The combination of former factors determines the speed at which new data in sufficient quantity can be produced whenever the need to adapt to novel conditions in either the generative model or the deep neural architecture arises. Accordingly, a crucial step is to identify a reasonable compromise between resource consumption and rendering fidelity. While a detailed analysis of the latter is out of scope of this work, we decided to use Epic Games' Unreal Engine 4 (UE4) [11] amongst other alternatives as it satisfies above mentioned requirements to the best of the authors' knowledge. A further aspect taken into consideration was the usability of the rendering software, specifically the open-source nature and underlying coding framework and interfaces, that permit modification of source code (For UE4 this is C++ code). While providing visually plausible images resulting from the underlying physically based shading [19], the real-time capabilities on current graphics processing units (GPU) are considered a substantial benefit.

We develop a rendering workflow, illustrated in Fig. 1, in consolidation with the priors derived in Sect. 2. Due to their constancy in placement and geometry,



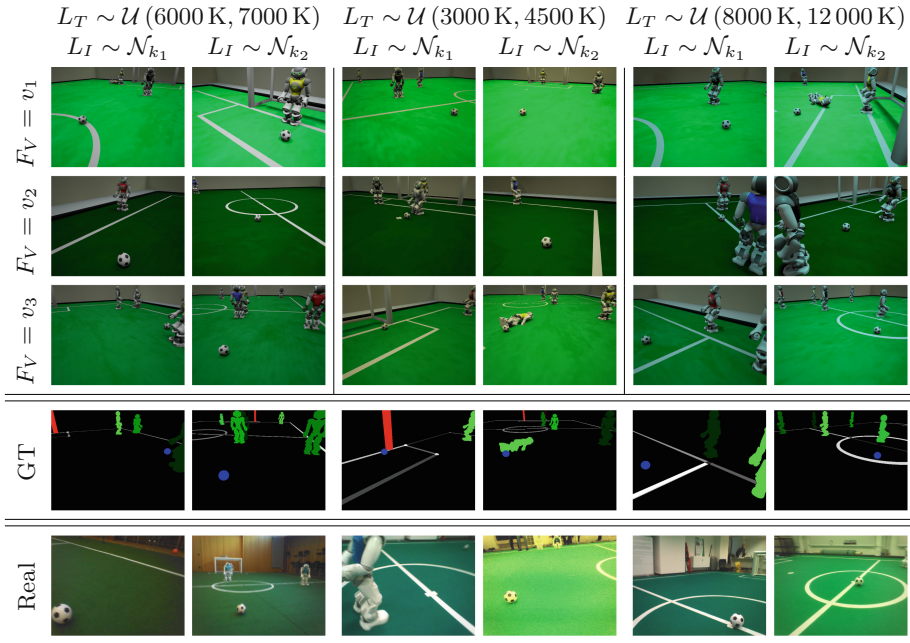
**Fig. 1.** The rendering workflow including the sampling processes of parameters. First, photometric parameters are used to set the scene, for which  $N_G$  different geometric configurations of objects are stochastically simulated. For each configuration an occlusion check is performed to ensure visibility of the ball before an image is rendered and the corresponding semantic ground truth segmentation mask is obtained. An overall amount of  $N_P$  scene configurations are sampled in this fashion.

the playing field as well as the goals are placed in a first step. A set of photometric parameters  $\Psi_P$  is sampled from the distributions described in Eqs. 5, 6 and 7 and corresponding scene attributes are set. For a given scene,  $N_G$  geometric configurations  $\Psi_G$  are drawn from the joint distribution of the probabilities given in Eqs. 1, 2, 3 and 4. For each such configuration, we cast a ray to perform a collision check to determine whether the ball is occluded by more than 50% by another object. If this occurs, sampling of  $\Psi_G$  is repeated. Otherwise the image is rendered and a semantically annotated ground-truth mask (GT) is created as explained below. This procedure is repeated for  $N_P$  distinct scene settings, each with  $N_G$  varying geometric configurations, resulting in an overall amount of  $N_P \times N_G$  annotated images. The respective object meshes were created using Blender [10], excluding the NAO robot which is provided by [21]. The full pipeline including parameters and their distributions has been exposed and can be modified by the user through UE4’s Blueprint framework (graphical user interface).

Apart from the evident benefit of saving time by making manual image annotation obsolete, automatic segmentation further guarantees bias- and error-free ground truth data. Objects of interest are selected from the list of entities present in the scene. For each pixel in the rendered image, we perform a ray-cast. If the first collision of the ray is with an object of interest, the respective index is written to the corresponding pixel in the segmentation mask. A file containing the mapping between indices and objects is generated.

Figure 2 showcases several example images using a discretized parameter set for ease of visualization. Three ranges of temperatures, reflecting white (6000–7000 K), yellow (3000–4500 K) and blue (8000–12000 K) tints that have empirically been observed to be most common in competition venues, are illustrated. Furthermore, two intensity-distributions  $\mathcal{N}_{k_1}$  and  $\mathcal{N}_{k_2}$  ( $\mu_{k_1} < \mu_{k_2}$ ) are depicted.

In the same spirit, three HSV-values with  $H = 120^\circ, S = 1.0, V \in \{v_1 = 0.85, v_2 = 0.45, v_3 = 0.65\}$ , corresponding to light, dark and medium bright green colors are depicted. Furthermore, respective ground truth segmentation masks for the third row images are shown. Real images have been included for qualitative comparison.



**Fig. 2.** Top three rows present a subset of rendered images drawn from the generative model. For ease of visualization, we chose images that correspond to discretized model parameters for the distributions of  $L_T, L_I$  and  $F_V$ . Ranges for  $L_T$  reflect white, yellow and blue color casts, the selected means of the intensity-distributions are equal to  $\mu_{k_1} = 1700 \text{ lm}$  and  $\mu_{k_2} = 3600 \text{ lm}$  with  $\sigma_{k_1} = \sigma_{k_2} = 50$ . The shown field colors represent green ( $H = 120^\circ, S = 1.0$ ) with different brightness values  $v_1 = 0.85, v_2 = 0.45$  and  $v_3 = 0.65$ . Semantic segmentation masks (GT) are visualized for the images in the third row. The bottom row shows real images for qualitative comparison. (Color figure online)

## 4 Deep Learning from Synthetic Images

To demonstrate the potential of our approach we evaluate deep convolutional neural networks in a classification context, where the training process is performed using synthetic images from our rendering workflow and accuracy is measured exclusively on real data. We consider a multi-class categorization comprised of the classes: robot, ball, goal post and field. The reason we do not include backgrounds outside the field boundaries stems from the assumption that common pre-processing steps are readily capable of identifying field boundaries.

Deep convolutional neural networks are typically trained using some form of stochastic gradient descent algorithms where the parameters  $\Theta$  of a (deep) neural network are optimized such that a loss function  $\mathcal{L}$  is minimized:

$$\Theta = \arg \min_{\Theta} \frac{1}{N} \sum_{n=1}^N \mathcal{L}(x_n, \Theta). \quad (8)$$

Here  $x_{1,2,\dots,N}$  denotes the training data set, and the optimization process is split into steps involving mini-batches  $x_{1,2,\dots,m}$  using estimates of the loss function’s gradient with respect to the network’s parameters. Using mini-batches is an often employed technique to speed up the optimization process and to introduce stochasticity into the gradient in order to let our network escape local minima [22]. In addition to mini-batches we apply momentum and weight decay term. The former in principle quickens learning convergence when gradients are aligned in subsequent steps, whereas the latter is a  $L_2$  regularization term. The interested reader is pointed to [23] for a detailed description of optimization methods and their subtleties.

#### 4.1 Data and Training Hyper-parameters

We derive a dataset of 25000 patches per class coming from an equivalent amount of unique scene configurations, without further augmentation. Here, a patch is defined as a rectangular image region spanning the area of an object and is extracted based on the semantically annotated mask, see Sect. 3 for details on how the mask is generated. A test set containing 780 patches per class has manually been extracted and annotated from real images taken in regular SPL game scenarios<sup>1</sup>. The training and evaluation of convolutional neural networks has been conducted using torch7 [24] on a single NVIDIA GTX 1080 GPU, deployment on the NAO robot has been realized by loading our trained networks with tiny-dnn [25] for optimized CPU usage. For our experimental evaluation of the neural networks we determine four suitable network structures with varying depths inspired by the works of [5,6] and chose a set of possible feature amounts in conjunction with runtime considerations on the current NAO hardware. For each of the four network structures the amount of features is determined by a parameter  $C_f$ , where  $C_f$  effectively represents a network’s representational capacity as the layers are defined to contain an amount of features equivalent to either  $2^{C_f}$  or  $2^{C_f+1}$  and  $C_f \in \{1, 2, \dots, 6\}$ . Consistent with [6] we express all “fully-connected” layers in the classifier through convolutions with spatial filter size  $1 \times 1$  both due to efficiency in computational implementation and accuracy [26]. All pooling layers compute a conventional max pooling operation. Each layer is furthermore followed by a Dropout [27] where 25% of a layer’s output units, or respectively 50% in fully-connected layers, is stochastically dropped. Activation functions are chosen to be Rectified Linear Units (ReLUs) [28], initialization

<sup>1</sup> We acknowledge contributions of parts of the real data generously provided by the following teams: HULKs, HTWK and SPQR.

follows the scheme proposed in [29] and cross-entropy has been used as a loss-function. One of the networks (BBN-M-C) replaces the fully-connected structure with a single convolutional layer without an activation function to map directly onto the classes similar to [6].

**Table 1.** CNN architectures with number of layers, spatial feature sizes and quantities, pooling dimensionality and added zero-padding. Architectures BBN-S and BBN-M are conceptually equal with the later having an extra layer and thus more representational complexity. BBN-M-C replaces the fully-connected structure with a single convolutional layer without an activation function to map directly onto the classes. BBN-L increases representational capacity through features with increased spatial size.

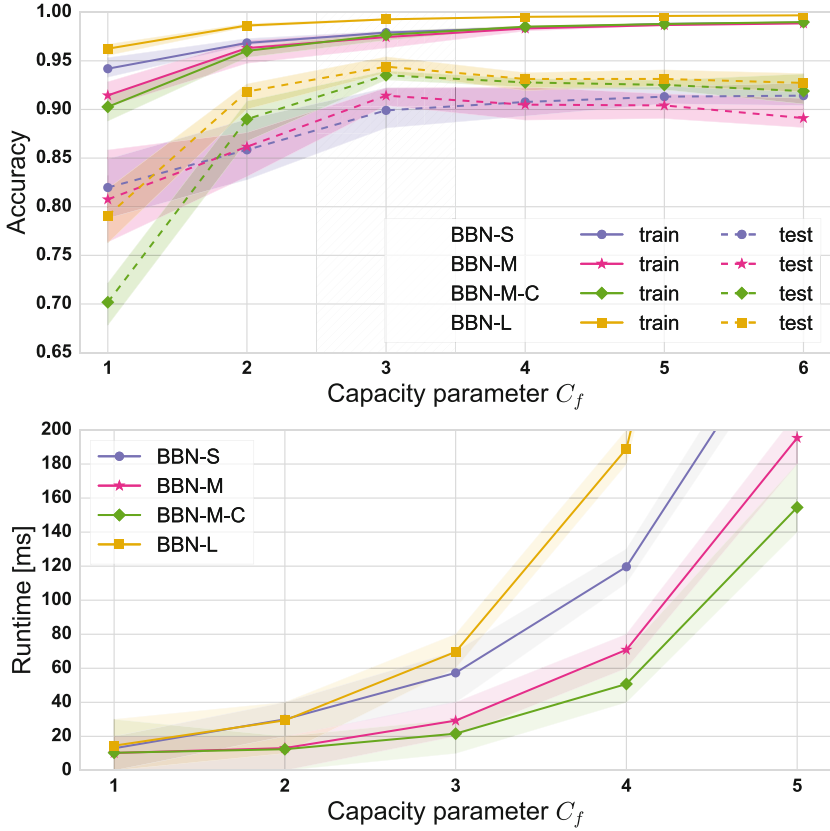
		BBN-S	BBN-M	BBN-M-C	BBN-L
Conv 1	No. features   Size	$2^{C_f}   5 \times 5$	$2^{C_f}   5 \times 5$	$2^{C_f}   5 \times 5$	$2^{C_f}   8 \times 8$
	Zero-padding	2	2	1	4
	Pooling size   Stride	$2 \times 2   2$	$2 \times 2   2$	$2 \times 2   2$	$4 \times 4   2$
Conv 2	No. features   Size	$2^{C_f+1}   3 \times 3$	$2^{C_f+1}   3 \times 3$	$2^{C_f+1}   3 \times 3$	$2^{C_f+1}   8 \times 8$
	Zero-padding	1	1	1	3
	Pooling size   Stride	$2 \times 2   2$	$2 \times 2   2$	$2 \times 2   2$	$4 \times 4   2$
Conv 3	No. features   Size	-	$2^{C_f+1}   3 \times 3$	$2^{C_f+1}   3 \times 3$	$2^{C_f+1}   5 \times 5$
	Zero-padding	-	1	1	3
	Pooling size   Stride	-	$2 \times 2   2$	$2 \times 2   2$	$2 \times 2   2$
F-Conv 1	No. features   Size	$256   1 \times 1$	$256   1 \times 1$	-	$512   1 \times 1$
	Stride, Padding	$1   0$	$1   0$	-	$1   0$
F-Conv 2	No. features   Size	$128   1 \times 1$	$128   1 \times 1$	-	-
	Stride   Padding	$1   0$	$1   0$	-	-
Output	No. features   Size	$4   1 \times 1$	$4   1 \times 1$	$4   3 \times 3$	$4   1 \times 1$

Hyper-parameters have been determined using a random search as presented in [30] on log-uniform scales with 20% of training data extracted uniformly and used for cross-validation. In particular the learning rate ( $10^0, 10^{-1}, \dots, 10^{-4}$ ), mini-batch size (16, 32,  $\dots$ , 128) and pre-processing methods (no pre-processing, zero-mean centering and global contrast normalization, see [23]) have been considered. Spatial input size of  $32 \times 32$ , a weight-decay of  $5 \cdot 10^{-4}$  and a momentum term of 0.9 are kept constant. We determined the following set of parameters that are used in all subsequent experiments: an initial learning rate of  $10^{-2}$ , a mini-batch size of 64 without any form of pre-processing. In addition to the initial learning rate we create a learning rate schedule, dividing the learning rate by a factor of 5 every 16 epochs, consistent with an observable plateau in our validation curve. With these parameters we trained for an overall of 40 epochs.

## 4.2 Network Accuracy, Capacity and Runtime Evaluation

We evaluate the influence of the representational capacity  $C_f$  on achieved accuracy and runtime for the previously determined hyper-parameters on the set





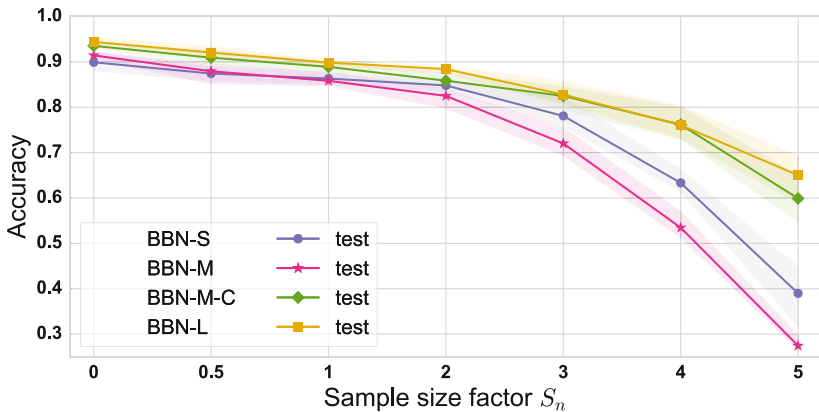
**Fig. 3.** Top panel: Train and test accuracies for the architectures defined in Table 1 in dependence on the capacity parameter  $C_f$  shown in a  $\log_2$ -uniform scale. Experiments were repeated five times for statistical consistency. Shaded regions represent minimum and maximum deviations from the obtained mean values. The hatched area depicts a (local) optimum of effective network capacity with under-fitting regimes for smaller and over-fitting present for larger  $C_f$  values. Bottom panel: Corresponding runtime on the NAO robot’s hardware for different  $C_f$ , evaluated and averaged on a thousand forward passes. The range is constrained to ensure that the area of interest (low runtimes) is adequately resolved and the overall trend (power law behavior) is clear.

of proposed neural network architectures. For statistical consistency we repeat network training and evaluation processes five times and report runtimes as the mean of a thousand forward passes. In the top panel of Fig. 3 corresponding train and test accuracies are illustrated, whereas respective runtimes can be found in the bottom panel.  $C_f < 3$  results in evident under-fitting, values greater than 3 seem to lie in a general over-fitting regime. While technically the test accuracy could plummet completely in this regime, the use of weight-decay counteracts this behavior resulting in only little loss in accuracy. In conjunction with the

evaluated runtimes on current NAO-H25 hardware it can be observed that all networks with  $C_f > 3$  improve neither accuracy nor runtime. For  $C_f = 3$ , the BBN-L network is able to achieve a best overall mean accuracy of 94.40% ( $\pm 0.6\%$ ). However with only 0.88% less accuracy and a mean runtime of 21.58 ms in contrast to 69.65 ms, the BBN-M-C network represents an applicable alternative regarding runtime requirements.

### 4.3 On Sampling Complexity

It remains an open question to what degree the sampling density influences achievable accuracy. For the presented task, we gain intuition and insights on the sampling density in our stochastic scene generation process for the given neural networks. From the originally generated training data set (25000 images per class) we repeatedly uniformly sample a fraction  $(2^{S_n})^{-1}$  of the initial quantity, where we refer to  $S_n$  as the sample size factor. Figure 4 shows the corresponding obtained accuracies. A clear correlation between sampling size and accuracy can be observed. Naturally, less data generally leads to worse performance.



**Fig. 4.** Accuracy of the proposed networks with  $C_f = 3$  on differently sized training sets. The sample size factor  $S_n$  determines a fraction  $(2^{S_n})^{-1}$  of the original training set size (25000 per class). Consistent with previous experiments, the mean accuracy of five repetitions is visualized. Shaded regions represent the deviations.

## 5 Conclusion

We developed a stochastic scene generation process for the RoboCup SPL, consisting of a generative model and synthetic image and semantically enriched ground-truth creation employing a state-of-the-art physically based rendering engine [11]. Compelling multi-class classification results on real-world data have been demonstrated on a variety of deep convolutional neural network architectures that have been trained entirely from 3-D simulation. The space of neural

network architectures, capacity, run-time and data quantity has systematically been probed, analyzed and insights have been shared. Our best network in terms of accuracy and speed is able to achieve approximately 94% accuracy in less than 22 ms per patch on current NAO-H25 hardware. Therefore the error-prone, tedious and time consuming manual human annotation and data gathering tasks have successfully been replaced.

Our approach provides the means for several future research prospects, that could include, but are not limited to: inferring the relative importance of individual scene parameters related to the image generation process (e.g. geometry, photometry, texture, etc.) for computer vision algorithms, the extension of the deep convolutional neural network based approach to detection approaches such as pixel-wise semantic image segmentation [16,31] or the general inclusion of further available information such as depth. Being modular in nature, the rendering workflow is furthermore readily extensible to generation of temporally coherent scenes for potential use in localization, navigation and motion-estimation tasks.

## References

1. 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. LNAI, vol. 9513, pp. 239–250. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-29339-4\\_20](https://doi.org/10.1007/978-3-319-29339-4_20)
2. Härtl, A., Visser, U., Röfer, T.: Robust and efficient object recognition for a humanoid soccer robot. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS, vol. 8371, pp. 396–407. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44468-9\\_35](https://doi.org/10.1007/978-3-662-44468-9_35)
3. 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](https://doi.org/10.1007/978-3-642-32060-6_13)
4. Qian, Y., Lee, D.D.: Adaptive field detection and localization in robot soccer. In: Behnke, S., Sheh, R., Sariel, S., Lee, D.D. (eds.) RoboCup 2016. LNCS, vol. 9776, pp. 218–229. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-68792-6\\_18](https://doi.org/10.1007/978-3-319-68792-6_18)
5. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A.C., Bengio, Y.: Maxout networks. *JMLR* **28**, 1319–1327 (2013)
6. Lin, M., Chen, Q., Yan, S.: Network in network. *CoRR* abs/1312.4400 (2013)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *CVPR*, pp. 770–778 (2016). <https://doi.org/10.1109/CVPR.2016.90>
8. Albani, D., Youssef, A., Suriani, V., Nardi, D., Bloisi, D.D.: A deep learning approach for object recognition with NAO soccer robots. In: Behnke, S., Sheh, R., Sariel, S., Lee, D.D. (eds.) RoboCup 2016. LNAI, vol. 9776, pp. 392–403. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-68792-6\\_33](https://doi.org/10.1007/978-3-319-68792-6_33)
9. Speck, D., Barros, P., Weber, C., Wermter, S.: Ball localization for robocup soccer using convolutional neural networks. In: Behnke, S., Sheh, R., Sariel, S., Lee, D.D. (eds.) RoboCup 2016. LNAI, vol. 9776, pp. 19–30. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-68792-6\\_2](https://doi.org/10.1007/978-3-319-68792-6_2)
10. Blender.org: Blender. <https://www.blender.org/>. Accessed 12 Mar 2017

11. EpicGames: Unreal engine 4. <https://www.unrealengine.com/>. Accessed 12 Mar 2017
12. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7577, pp. 611–625. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33783-3\\_44](https://doi.org/10.1007/978-3-642-33783-3_44)
13. Fischer, P., et al.: FlowNet: learning optical flow with convolutional networks. In: ICCV (2015)
14. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: ground truth from computer games. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 102–118. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46475-6\\_7](https://doi.org/10.1007/978-3-319-46475-6_7)
15. Lerer, A., Gross, S., Fergus, R.: Learning physical intuition of block towers by example. In: ICML, vol. 48, pp. 430–438 (2016)
16. Veeravasaram, V.S.R., Rothkopf, C.A., Ramesh, V.: Model-driven simulations for deep convolutional neural networks. CoRR abs/1605.09582 (2016)
17. RoboCup@Home Technical Committee: Robocup rulebook. <http://www.tzi.de/spl/pub/Website/Downloads/Rules2016.pdf>. Accessed 14 Mar 2017
18. SoftbankRobotics: NAO H25 technical specifications. [http://doc.aldebaran.com/2-1/family/nao\\_h25/index\\_h25.html#nao-h25](http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html#nao-h25). Accessed 26 Mar 2017
19. Karis, B., Games, E.: Real shading in unreal engine 4. In: Physically Based Shading Theory and Practice, SIGGRAPH (2013)
20. docs.unrealengine.com: Lighting basics. <https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Basics/>. Accessed 26 Mar 2017
21. SoftbankRobotics: NAOmodel. <https://community.ald.softbankrobotics.com/en/resources/software/language/en-gb>. Accessed 14 Mar 2017
22. Wilson, D., Martinez, T.R.: The general inefficiency of batch training for gradient descent learning. *Neural Netw.* **16**, 1429–1451 (2003). [https://doi.org/10.1016/S0893-6080\(03\)00138-2](https://doi.org/10.1016/S0893-6080(03)00138-2)
23. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
24. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: a matlab-like environment for machine learning. In: BigLearn, NIPS Workshop (2011)
25. Zheltonozhskiy, E.: Tinydnn. <https://github.com/tiny-dnn/tiny-dnn> (2017). Accessed 12 Mar 2017
26. Springenberg, J., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net. In: ICLR (Workshop Track) (2015)
27. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *JMLR* **15**, 1929–1958 (2014)
28. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: *JMLR, AISTATS 2011*, vol. 15, pp. 315–323 (2011)
29. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *JMLR, AISTATS 2010*, vol. 9, pp. 249–256 (2010)
30. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *JMLR* **13**, 281–305 (2012)
31. 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. CoRR abs/1606.00915 (2016)