Designing Convolutional Neural Networks Using a Genetic Approach for Ball Detection

Georg Christian Felbinger Patrick Göttsch Pascal Loth Lasse Peters Felix Wege

RobotING@TUHH e.V., Hamburg University of Technology

HULKs — RoboCup SPL Team • TU Hamburg, Germany

Introduction

In 2016, a black and white patched ball was introduced into the Standard Platform League (SPL). Requirements for a detection algorithm comprise a robust detection and classification in dynamically changing environments, as well as a cost-effective real-time computation on the NAO.

Approaches based on convolutional neural networks (CNN) for object detection led to promising results in RoboCup SPL [1]. However, hyperparameters for the structural setup of such networks need to be chosen carefully. Genetic approaches can be used to determine an optimized network topology. [2] described the evolution of fully connected network topologies. The idea can easily be applied to other model components, e.g. convolutional layers. A similar genetic approach was used by [3] to automatically discover good architectures of CNNs.

This paper presents a genetic framework to design CNNs for real-time applications on computationally weak hardware by simultaneously optimizing the classification performance and inference complexity. Our approach considers a bounded capability to collect large amounts of training data and allows the user to prioritize true negative rate and true positive rate suitable for a specific task. The detection of a black and white ball on the NAO robot is used to demonstrate the performance of the framework.

Data

Due to the limited computation capabilities of the NAO, not the whole image can be inferenced by a neural network. Therefore, a deterministic region of interest is needed to provide candidate regions to be classified.

Figure 1: Seed is corresponding to the center of the black patches on the ball.

Figure 2: Merged seeds and projection of the corresponding ball radius.

Figure 3: Reprojected ball from result of the ball filter (green circle bounded black rectangle).

Data Setup

The data used for training and evaluation of the classifier were collected during various events (RoboCup 2017, Iran Open 2017, German Open 2017, weekly test games).

It consists of 16880 positive examples (candidate images containing a ball) and 23876 negative examples (candidate images not containing a ball).

During training negative examples are subsampled randomly to ensure that in every cross-validation set the same amount of positive and negative examples are present.

Genetic Design of CNNs - Search Space

The input is a YCbCr candidate image of arbitrary quadratic size. It is resized to a fixed quadratic size using nearest neighbor interpolation. Then, multiple convolutional layers are applied. The next step is a batch normalization layer. Finally, multiple fully connected layers are applied. The output is a vector representing the class scores.

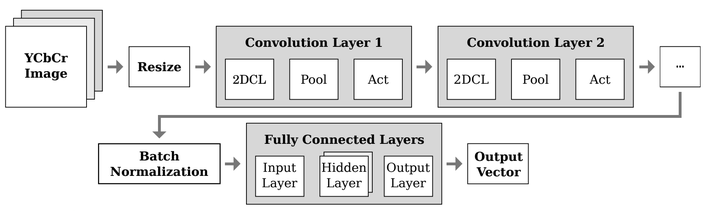


Figure 4: General structure of a CNN. Each convolutional layer consists of a two dimensional convolution mask (2DCL) followed by a pooling layer (Pool) and an activation function (Act). The convolved and normalized image is fed into multiple fully connected layers yielding the final output vector.

Each individual specifies the remaining hyperparameters within this structure. These are the input size, number of convolutional and fully connected layers as well as their internal configuration. Each convolutional layer is parameterized with a mask size, pooling type and activation function. Likewise, the parameters of a fully connected layer consists of the size and activation function.

Table 1: Search space with value ranges derived from the general CNN structure.

Parameter		Value Range
Input	Sample Size	$[8, 16]$
Convolutional	Layers	$[0, 2]$
	No. of Masks	$[1, 5]$
	Mask Size	$[2, 3] \times [2, 3]$
	Activation	Tanh, ReLU
	Pooling	None, Avg, Max
Fully Connected	Layers	$[0, 4]$
	Size	$[2, 20]$
	Activation	Tanh, ReLU

Genetic Design of CNNs - Fitness Function

We optimize classification performance and inference complexity at the same time. In the fitness function classification performance is represented by the true negative and true positive rate. Inference complexity is approximated asymptotically.

Classification Performance

For each network a k-fold cross validation was performed which yielded k values for true negative rate TNR_{nk} and true positive rate TPR_{nk}. In order to approximate a lower bound of these performance metrics the difference of mean and variance were used in the fitness function. The TPR_n and TNR_n for a network n was computed by:

$$TPR_n = \text{Avg}(TPR_{n1}, ..., TPR_{nk}) - \text{Var}(TPR_{n1}, ..., TPR_{nk}) \quad (1)$$

$$TNR_n = \text{Avg}(TNR_{n1}, ..., TNR_{nk}) - \text{Var}(TNR_{n1}, ..., TNR_{nk}) \quad (2)$$

where Avg is the arithmetic mean and Var is the variance.

Inference Complexity

The complexity of a network was asymptotically approximated and linearly scaled. The complexity cc of a convolutional layer i is approximated by equation (3).

$$cc_i = \frac{I_x \cdot I_y \cdot I_c \cdot m_x \cdot m_y \cdot m_c}{\hat{I}_x \cdot \hat{I}_y \cdot I_c \cdot \hat{m}_x \cdot \hat{m}_y \cdot \hat{m}_c} = \frac{I_x \cdot I_y \cdot m_x \cdot m_y \cdot m_c}{\hat{I}_x \cdot \hat{I}_y \cdot \hat{m}_x \cdot \hat{m}_y \cdot \hat{m}_c} \quad (3)$$

Symbols I_x, I_y, I_c correspond to layer input size and depth, m_x, m_y, m_c to amount and size of the convolution masks in this layer. While $\hat{I}_x, \hat{I}_y, \hat{m}_x, \hat{m}_y, \hat{m}_c$ represent maximum values as defined by the search space.

The complexity cf of the fully connected part is approximated by equation (4).

$$cf = \frac{\sum_{i=1}^{k} s_i \cdot s_{i-1}}{\sum_{i=1}^{k} \hat{s}_i \cdot \hat{s}_{i-1}} \quad (4)$$

The number of hidden layers is denoted by k and the size of layer i by s_i. The input vector size is s_0.

Hence, the final complexity of a network topology with j convolutional layers is

$$c_n = 1 - \frac{\sum_{i=1}^{j} cc_i + cf}{j + 1}. \quad (5)$$

Resulting Fitness Function

Given the approximation of classification performance and inference complexity the resulting fitness function is chosen as follows:

$$f_n = 0.7 \cdot TNR_n^2 + 0.25 \cdot TPR_n^2 + 0.05 \cdot c_n. \quad (6)$$

For networks with a good classification performance it is disproportionally difficult to further increase the TNR and TPR. Thus, the TNR and TPR are squared in the fitness function.

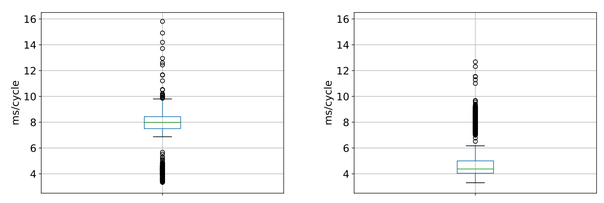
Evaluation on NAO

Generalization Test

The best network of the last generation that was trained with all of the training data is subjected to a final generalization test. This final network is evaluated with data collected in another environment which can be considered to be a proper generalization test because no data from these testing conditions was used during training. The classifier predicted 4989 of 5687 positives and 12680 of 12730 negatives correctly resulting in a $TNR = 0.99$ and a $TPR = 0.87$.

Runtime Analysis on the NAO Robot

The whole ball detection including the resulting network running on the NAO was evaluated. For this test we fixed the number of generated candidates per image to the average amount five to get stable measurement results. Figure 5 shows the result of those measurements. With an average runtime of about 8ms on the top and 4ms on the bottom camera we reached our real-time criteria which is 30ms for a vision cycle.



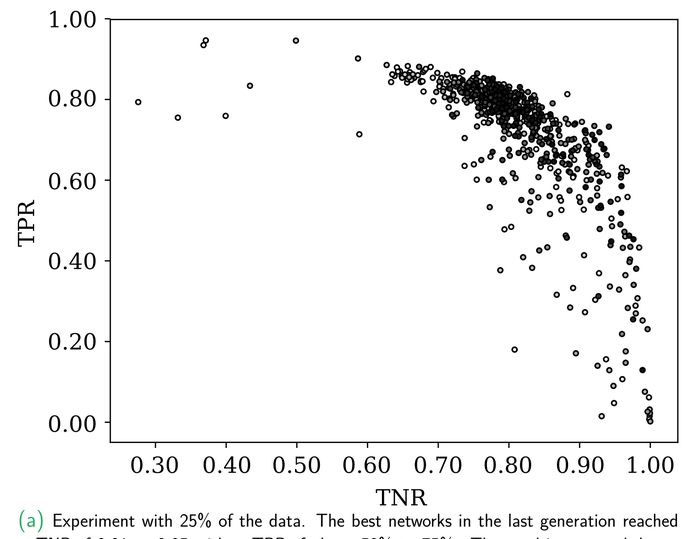
(a) Top Camera (b) Bottom Camera

Figure 5: Runtime of the ball detection including the resulting network on the NAO. The green line indicates the mean of the runtime. The interquartile range is shown by the blue box. The upper black bar illustrates the 0.75-quantile, respectively the lower black bar the 0.25-quantile. The circles correspond to outliers.

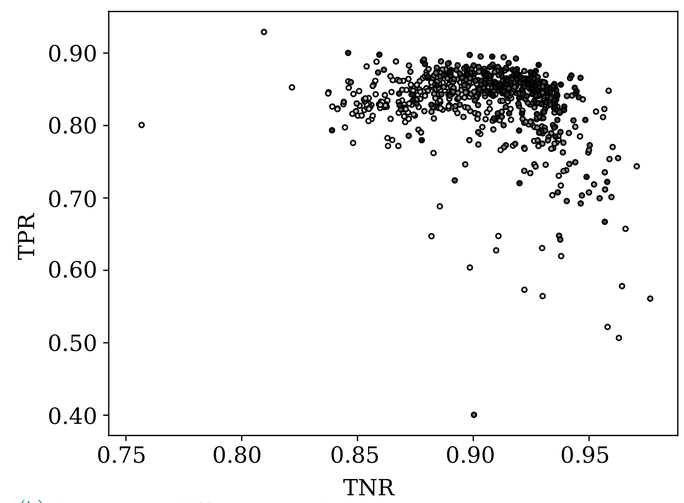
Experiments and Evaluation

In every experiment 15 generations with 50 networks in each generation were evaluated. The worst 10% in each generation were excluded from reproduction. The mutation probability was set to $\frac{1}{16}$ according to the maximum number of degrees of freedom of the given search space.

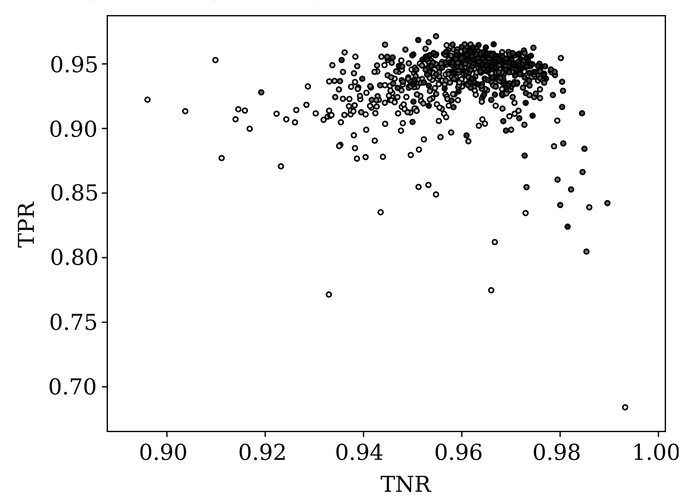
The algorithm should be able to design a CNN as a solution to the ball detection problem considering a limited amount of training data. To show that this can be achieved three experiments were conducted, sampling 25%, 50% and 100% of the available training data.



(a) Experiment with 25% of the data. The best networks in the last generation reached a TNR of 0.91 to 0.95 with a TPR of about 50% to 75%. The resulting network has a very small sample size of 8×8, one convolutional layer of four masks and only three hidden layers in the fully connected part.



(b) Experiment with 50% of the data. The best networks in the last generation reached a TNR of about 0.93 with a TPR above 0.85. Networks with one large convolutional layer and mainly three hidden layers in the fully connected part dominated this experiment.



(c) Experiment with 100% of the data. The best networks in the last generation reached a TNR of about 0.95 with a TPR above 0.90. While the sample size and convolutional layers remained similar to those in the second experiment, the fully connected part converged to four hidden layers instead of three.

Figure 6: Evolution of classification performance. Note that scaling differs between experiments as the overall results got better. Results of the first generation are plotted with white filled circles. Results of the following generations are plotted in increasingly darker shades of gray.

The network having the highest score in the last generation is considered to be the resulting network. Networks became more complex while increasing the amount of data resulting in better classification performance.

Conclusion

We presented a genetic framework that was successfully applied to the problem of black and white ball detection using little computational power. Our experiments showed that a genetic approach is able to identify a small yet efficient network suitable for a specific classification task. The presented optimization strategy obtains suitable hyperparameters even with limited amount of training data.

In order to enhance convergence speed future work should focus on evaluating different variants of genetic algorithms such as elitism [4]. Additionally, we would like to apply the approach to multiclass problems using a modified fitness function.

References

[1] J. Menashe *et al.*, "Fast and precise black and white ball detection for robocup soccer," in *Robot world cup*, 2017.

[2] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[3] Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks for image classification," *CoRR*, vol. abs/1710.10741, 2017.

[4] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," Carnegie Mellon University, Pittsburgh, PA, CMU-CS-95-141, May 1995.