International Journal of Robotic Computing

Real-time Driving Context Understanding using Deep Grid Net: A Granular Approach

Liviu A. MARINA
ROVIS Lab (Robotics Vision and Control), Transilvania University of Brasov, Brasov, 500036, Romania
marina.liviu.alexandru@unitbv.ro
http://www.rovislab.com

Florin D. MOLDOVEANU
Departament of Automation, Transilvania University of Brasov, Brasov, 500036, Romania
moldof@unitbv.ro

Sorin M. GRIGORESCU
ROVIS Lab (Robotics Vision and Control), Transilvania University of Brasov, Brasov, 500036, Romania
Elektrobit Automotive
s.grigorescu@unitbv.ro
http://www.rovislab.com

Numerous self-driving cars algorithms rely on grid maps for motion planning, obstacles avoidance or environment perception. Obtained from fused sensory information, the occupancy grids (OGs) are nowadays among the most popular solutions used in series production in automotive industry. In this paper, we extend *Deep Grid Net* (DGN) [1], a deep learning(DL) system designed for understanding the context in which an autonomous car is driving. We consider this paper a granular approach to DGN method due to the improvements added to the original research [1]. DGN incorporates a learned driving environment representation based on OGs obtained from raw real-world Lidar data and constructed on top of the Dempster-Shafer (DS) theory. Our system is able to predict in real-time if the vehicle is driving on highway, on county roads, inside a city, in parking lots or is stuck in a traffic jam. The predicted driving context is further used for switching between different autonomous driving strategies implemented within EB robinos, Elektrobit's Autonomous Driving (AD) software platform. We propose a neuroevolutionary approach to search the optimal hyper-parameters set of DGN. Genetic algorithms (GAs) were selected due to their demonstrated capabilities to evolve deep neural networks with improved accuracy and processing speed. The performance of the proposed deep network has been evaluated against similar competing driving context estimation classifiers.

Keywords: Deep Learning; Genetic Algorithms; Occupancy Grids.

1. Introduction

An essential requirement for autonomous vehicles is the ability to perceive and understand in real-time the surroundings in order to instrument a responsive answer. An important amount of resources were allocated in the last years to solve the environment understanding

challenges. Neural networks become the main solution for analysis of the considerable amount of data coming from the various sensors mounted on vehicles. Surveys dedicated to autonomous vision and environment perception can be found in [2] and [3]. Autonomous driving in urban street scenarios achieved a strong progress recently. In order to navigate autonomously in complex driving scenarios a vehicle demands a deep scene understanding.

The context in which the vehicle is driving represents an important input for a *Highly Autonomous Driving* (HAD) systems in order to deploy the ideal driving strategies when driving within a city in comparison with driving on a country road. There are several use cases in which the driving situation classification can represent an important source of information for a vehicle behaviour arbitration mechanism. The missing of *Global Positioning Systems* (GPS) signal or the low accuracy can be one of the use cases.

This paper is an extended version of the work published in [1] where *Deep Grid Net* (DGN) algorithm, illustrated in Fig. 1 was introduced. This work is a granular approach aiming to add more implementation details and experiments results to our original research. DGN algorithm predicts the driving context by analyzing local *Occupancy Grids* (OGs) constructed from fused raw sensory data. The OGs are preferred over images due to the highly reduced search space determined by the lower information representation. The OGs describe the driving environment as free or occupied spatially distributed cells, as later explained in Section 3. As in the previous work [1], the occupancy grids are built from data acquired from Lidar sensors, mounted on the front and rear sections of the ego-car.

The DGN approach leverages on the power of deep neural architectures for learning a grid-based representation of the traffic scene. By using OGs instead of raw image data, we are able to cope with uncertainties present in the driving scenes, such as changes in the sensors calibration, pose, time and latency. This learned representation can be used in different autonomous driving tasks which rely on driving context understanding.

The DGN algorithm is deployed within Elektrobit's HAD software framework, coined EB robinos. A brief description of EB robinos is given in Section 4. DGN offers a robust real-time estimation of the driving context mapped to five classes: driving on the highway, driving in the inner-city, driving on the country roads, driving in ares with traffic jam situations and parking. This information is further used by the Behavior Arbitration layer of the framework for planning different autonomous driving strategies.

Grid Maps, or *Occupancy Grids* (OGs), have their origins in robotic applications [4], [5]. A traditional grid map divides the environment into single grid cells and estimates the occupancy probability of each cell. We have used grid information as data source for our deep network, thus obtaining a bird's-eye view perspective of the traffic scene, as shown in the images from Fig. 2. Each cell is color-coded, red pixels representing obstacles, green marking the free space and black representing the unknown occupancy. Apart from the free/occupied information, the grid also encodes the occupancy confidence, also referred to as occupancy probability, or *occupancy belief*.

Deep convolutional neural networks (DCNNs) were chosen to encode the traffic scene due to their generalization capabilities. DCNNs performance depends on many hyper-parameters, namely loss functions, optimization algorithms and the neurons number from different layers. In order to overcome the manually tuning of these hyper-parameters, we

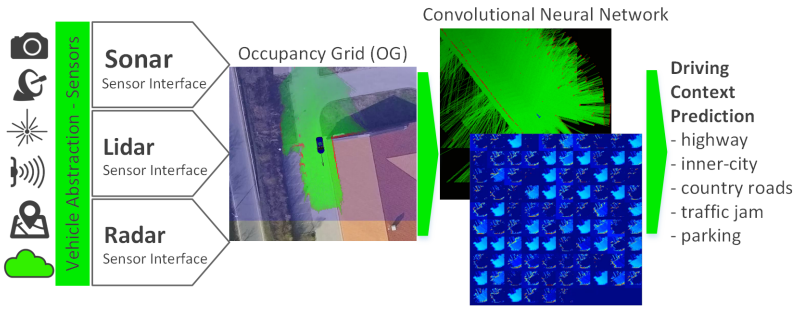

Fig. 1. Deep Grid Net (DGN) architecture. Lidar sensory data streams are converted into Occupancy Grid (OGs), which are furthered parsed by a Convolutional Neural Network. The final network layer provides the driving context as a five classes probabilistic output, that is, driving on the highway, driving in the inner-city, driving on the country roads, driving in ares with traffic jam situations and parking. A video which demonstrates the DGN's functionality can be found following this link

build on top of the authors previous work on one-shot learning using neuroevolutionary algorithms [6] and propose an approach for the automatic computation of hyper-parameters during training.

The main contributions of this paper can be summarized as follows:

- Improvement of Deep Grid Net (DGN) classification accuracy;
- Extension of DGN's hyper-parameters set which are optimized using Genetic Algorithms (GAs);
- Deployment and evaluation of DGN into the EB robinos autonomous driving software stack.

The rest of the paper is organized as follows: an overview of related work is given in Section 2, while the *Deep Grid Net* system is presented in Section 3. A description of EB robinos, the training strategy, and the evaluation of DGN's performance are given in Section 4. Finally, the conclusions are stated in Section 5.

2. Background and Motivation

Occupancy grids are widely used in robotics field for autonomous navigation of self-driving agents. In [7], *Convolutional Neural Networks* (CNNs) have been trained on 2D range sensory data for the semantic labeling of places in unseen environments. This approach allows a robot to use Lidar for space classification, where OGs created from Lidar scans are converted to gray images used for training the CNN. In [7], the robot is able to distinguish between three classes, that is, room, corridor, and doorway, which are used to create a localization map. Although similar, this solution to indoor mapping does not apply to outdoor autonomous driving, where the traffic scene has a more complex structure. Driving through an outdoor environment implies the interaction with dynamic objects, an interaction which is not taken into consideration by the method presented in [7]. In our work, this drawback

is eliminated by the OG degradation property, described in Section 3.

Recurrent Neural Networks (RNN) have been used by Ondruska [8] for tracking and classifying the surroundings of a robot placed in a dynamic and partially observable environment. A recurrent neural network filters the input stream composed of raw laser measurements in order to infer the objects locations together with their identity, in both visible and occluded areas. The algorithm in [8] takes inspiration from Deep Tracking [9], which is a deep learning system that leverages on deep neural networks (DNNs) for end-to-end tracking. Raw sensory data is used to construct an occupancy grid where the visible pixels are labeled for a classifier in a supervised manner. The training data has been recorded from a static and stationary position of the robot, resulting in low data variability. In order to cover as many driving scenes as possible, we have used in our system Lidar data which has been collected from various dynamic traffic scenarios. This procedure ensures a high sensory data variation.

In [10], an environment modeled with a Bayesian filtering technique is processed through a DNN, with the purpose of obtaining a long-term driving situation prediction for intelligent vehicles. This work is based on the principles stated by Nuss in [11], [12], where raw Radar and Laser data is parsed through a fusion layer. The algorithm predicts future static and dynamic objects using a CNN trained on occupancy grids. As we have also established, the filtered representation provided by OGs is a robust alternative to raw image data, thus increasing the accuracy of driving context classification.

Although OGs are common tools in robotics, there are only a few cases where deep learning techniques are used for real-time environment perception. This niche has not been sufficiently studied and can present many research possibilities. In [13], an improved version of the *Proportional Conflict Redistribution* rule '#'6 (PCR6), taking into account Zhang's degree of intersection of focal elements [14], was used on Lidar data. The estimated occupancy grids obtained from different methods, like PCR6 and Dempster-Shafer (DS) [15], are very similar to the ones used in our work.

In order to improve environment perception using OGs, object detection and classification can add important features, as shown in [16]. CNNs were applied on occupancy maps for encoding 3D range sensory information. Given as input this environment representation, the object detector outputs a list of oriented shapes and their corresponding semantic labels.

In [17], OGs and DNNs have been applied to outdoor driving scene classification. A major differences with respect to our work is that the classifier in [17] estimates only four driving classes based on OGs which are constructed by accumulating data over time. Only after accumulating enough information, the OGs can be used for classification. In our work, we compute OGs in real-time, during the movement of the vehicle. DGN also aims to increase the granularity of classification by taking into account five road types.

To improve the driving context prediction performance, a fine-tuning of the hyper-parameters set used in the DGN's training process is needed. GAs are used to avoid the manual parameters fine-tuning, based on multiple full training trials. Several papers demonstrated the efficiency of using GAs for learning the optimal hyper-parameters set. In [18] the authors present the advantages of using GAs for automatic selection and tuning of the

hyper-parameters over various techniques. The algorithm introduced in [19] uses ideas like mating and mutation in order to help the DNN architecture to learn to optimize its hyper-parameters by itself rather than have the values explicitly set. As it was demonstrated in [20], GAs can be applied even for hard deep reinforcement learning problems, evolving networks with over four million free parameters, the largest neural networks ever evolved with a traditional evolutionary algorithm.

The evaluation of a given hyper-parameters set can take a considerable amount of time and the search space is often very high-dimensional. In [21] the authors propose a solution to speed up the evaluation of DCNNs, solution that can be applied for our classification algorithm.

3. Methodology

3.1. *Problem space*

The Deep Grid Net algorithm is mainly composed of three elements: (i) an occupancy grid fusion algorithm, (ii) a deep neural network used for parsing the OG in real-time and (iii) an evolutionary algorithm used for selecting the optimal neural network set. The outcome obtained from DGN is a driving context classification, mapped to five classes: inner city, country road, parking lot, highway and traffic jam.

The OGs training dataset D is used to calculate the optimal DGN hypothesis h_{DGN}, which encodes the deep network's structure and weights. We define our problem space within the following Bayesian framework:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{1}$$

where $P(h)$ is the prior probability over h, $P(D) = \int_h P(D|h)P(h)$ is the training data probability, independent of h and $P(h|D)$ is the likelihood of h given D. $P(D|h)$ is the data likelihood over a given hypothesis.

The maximum a posteriori (MAP) hypothesis h_{MAP} can be defined as

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(h|D). \tag{2}$$

Using Bayes theorem, Eq. (2) can be written as:

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)}, \tag{3}$$

where $P(D)$ is not influencing the maximization problem, reducing Eq. (3) to:

$$h_{MAP} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(D|h)P(h). \tag{4}$$

Assuming that all hypotheses are equally probable, we can choose a *Maximum Likelihood* (ML) approach for training the deep network:

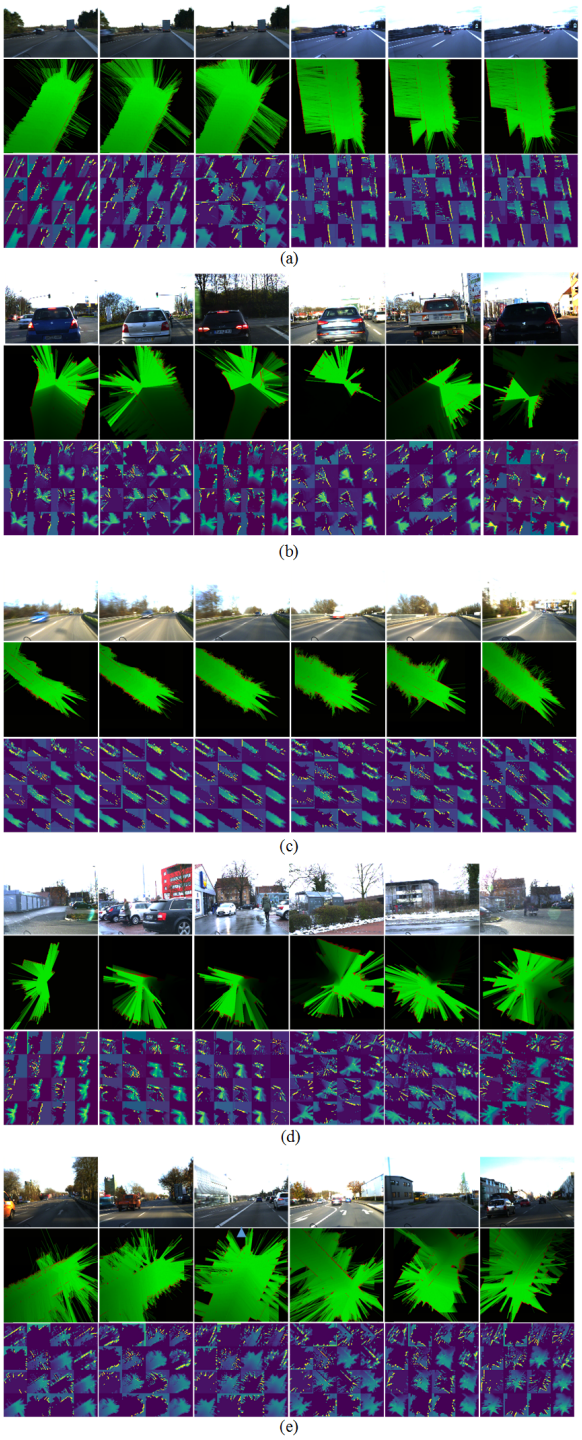

Fig. 2. Samples of real-world occupancy grids. The top images in each group represent snapshots of the driving environment, together with their respective OG and the activations of the first convolutional layer of DGN. The OG are obtained using data from a Lidar sensor mounted on top of our test vehicle. DGN is able to distinguish between highways (a), traffic jams (b), country roads (c), parking lots (d) and inner-city streets (e).

$$h_{ML} = \underset{h \in \mathscr{H}}{\operatorname{argmax}} P(D|h) = \underset{h \in \mathscr{H}}{\operatorname{argmax}} L(h) \tag{5}$$

The training samples are considered to be independently identically distributed, thus satisfying the following statement:

$$P(D|h) = \prod_{i=1}^{m} P(\langle x_i, y_i \rangle | h) = \prod_{i=1}^{m} P(y_i|x_i;h)P(x_i) \tag{6}$$

Maximizing the Eq. (6) is equivalent with the maximization of the logarithmic function $\log L(h)$:

$$\log L(h) = \sum_{i=1}^{m} \log P(y_i|x_i;h) + \sum_{i=1}^{m} \log P(x_i) \tag{7}$$

The term $\sum_{i=1}^{m} \log P(x_i)$ depends on D, but not on h. Hence, it can be ignored when searching for the optimal DGN hypothesis:

$$\log L(h_{DGN}) = \sum_{i=1}^{m} \log P(y_i|x_i;h) \tag{8}$$

3.2. *Occupancy Grids*

Occupancy Grids are the data source for calculating the optimal DGN hypothesis h_{DGN}. OG are often used for environment perception and navigation, applications which require techniques for data fusion and obstacles avoidance. In our work, the grids used for driving context classification where built using the *Dempster-Shafer* (DS) theory, also known as the *Theory of Evidence* or the *Theory of Belief Functions*, developed by Shafer in 1976 from Dempster's work [15]. It is characterized by four functions: a *Frame of Discernment* (FoD) function, a *Belief* (Bel) function, the sources of evidence represented by a *Basic Belief Assignment* (BBA) and the *Plausibility* (Pl) function. In the followings, a short introduction to belief functions and fusion rules will be provided for a better understanding on how the OG is built using the DS method. A pedagogical example of the DS approach chose to build our OGs is illustrated in Fig. 3.

The theory of belief functions is a general framework for reasoning with uncertainty, with connection to other frameworks such as possibility and probability theories.

Let Ω be a finite discrete FoD:

$$\Omega = \{w_1, w_2, w_3, \ldots, w_n\} \tag{9}$$

where $n > 1$ for the considered fusion problem and its fusion space G^{Ω}. D^{Ω} is the hyper-power set, or the super-power set S^{Ω}, depending on the considered problem [22]. A *Basic*

Belief Assignment (BBA) associated with the given set of evidence is defined as the mapping:

$$m(.) : G^{\Omega} \rightarrow [0,1] \tag{10}$$

satisfying $m(\Phi) = 0$ and $\Sigma_{A \in G^{\Omega}} m(A) = 1$, where $m(A)$ is the mass of belief of A committed by the source of evidence. The fusion space G^{Ω}, also named the power set, is the set of all discernment frames, that is, all hypotheses and all possible unions of hypotheses. The hyper-power set D^{Ω} and the super-power set S^{Ω} are defined as the set formed by all unions and intersections of hypotheses, or set formed by all unions, intersection, and complements of the hypotheses, respectively [23].

The *Belief* (*Bel*) and *Plausibility* (*Pl*) functions are defined as:

$$Bel(A) = \sum_{\substack{B \subseteq A \\ B \in G^{\Omega}}} m(B), \quad Pl(A) = \sum_{\substack{B \cap A \neq \phi \\ B \in G^{\Omega}}} m(B) \tag{11}$$

The maximum amount of potential specific support that could be given to A is quantified by the degree of plausibility $Pl(A)$, while the amount of justified support that should be given to A is quantified by the degree of belief $Bel(A)$ for a subset A. If for some $A \in G^{\Omega}$, $m(A) > 0$ then A is called a focal element of BBA $m(.)$. If all focal elements are singletons and $G^{\Omega} = 2^{\Omega}$, then BBA $m(.)$ is called a Bayesian BBA [15] and its belief function $Bel(A)$ is homogeneous and is equal to $Bel(A) = Pl(A)$. Otherwise $Bel(A) \leq Pl(A)$.

From the different fusion rules proposed in literature [22], the DS rule was most suited for our work, being also one of the most common rules used in the theory of evidence. The issue which arises here is how to combine two independent sets of probability mass assignments with specific situations. This rule derives commonly shared beliefs between multiple data sources and ignores all of the conflicted beliefs through a normalization factor. The joint mass is calculated from the $m_1(.)$ and $m_2(.)$ sets of masses. The DS combination is defined by taking $m_{1,2}^{DS}(\Phi) = 0$ for all $X \neq \Phi$:

$$m_{1,2}^{DS} \triangleq \frac{1}{1 - m_{1,2}(\phi)} \sum_{\substack{X_1, X_2 \in 2^{\Omega} \\ X_1 \cap X_2 \neq \phi}} \prod_{i=1}^{2} m_i(X_i) \tag{12}$$

where $m_{1,2}(\phi)$ measures the amount of conflict between the two mass sets. The term $1 - m_{1,2}(\phi)$ is a normalization constant. The total degree of conflict between the two sources of evidence is defined as:

$$m_{1,2}(\phi) \triangleq \sum_{\substack{X_1, X_2 \in 2^{\Omega} \\ X_1 \cap X_2 = \phi}} \prod_{i=1}^{2} m_i(X_i) \tag{13}$$

According to Shafer [15], the combination rule cannot be applied when two sources are in total conflict. It is said that the two sources are in total conflict if the condition

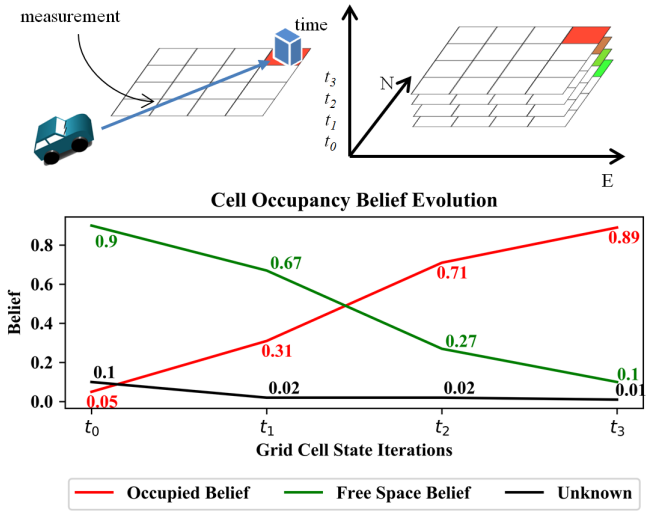

Fig. 3. Pedagogical example illustrating the behavior of the DS algorithm. The ego-car encounters an obstacle when driving on the North-East (NE) direction. Initially, at time t_0, the cell is marked as free space (green color). Once the car approaches an obstacle, the respective cell's occupancy belief increases, shown here by the red marking at time stamp t_3.

$m_{1,2}(\phi) = 1$ applies. Additional details on Dempster's *Rule of Combination* can be found in [24].

The occupancy grids provide a birds-eye perspective on the traffic scene. The basic idea behind OGs is the environment's division into 2D cells, each cell representing the probability, or belief, of occupation. Each cell is color-coded, green pixels representing the free space, red marking the occupied cells (or obstacles) and black models the unknown occupancy. The color intensity represents the degree of occupancy. The higher the intensity of the green color is, the higher the probability of a cell to be free is.

In this work, the content of the grid layer gets degraded over time by gradually decreasing the occupancy information for every grid cell. The grid content is updated over and over again, in real-time, with each sensory measurement.

The range of the sensors and the degrade factor were used to define the OG's dimension and precision. The world coordinate system is the reference for the OGs, sliding along the ego vehicle's driving direction. An anchor describing the position of the grid along the driving direction is used when new data is filled inside the occupancy map.

In order to have a better viewing perspective of the traffic scene, the ego-vehicle should always be centered in the grid. Due to the different data acquisition timestamps, this is not always the case. This phenomenon happens because the viewing range of the sensor is higher than the distance between the car and the outer grid cell. This inconvenience may cause the vehicle to move around 15m inside the grid, without moving the grid information itself. When newly received sensory data falls outside of the current grid system, the anchor

is moved and the grid switches position in a forward direction.

Due to the different sampling times, this behavior is potentially dangerous for the OGs consistency. As solution, we continuously move the grid relative to the position of the ego vehicle, while the anchor is tracked in parallel for grid data integration.

The OGs used for training are labeled through the visual inspection of the camera image provided by the video sensor. Examples of labeled OGs are shown in Fig. 2.

3.3. *Neuroevolutionary Training of DGN*

The optimization problem of the hyper-parameters set can be briefly defined as the amount of function evaluations that will be performed on every optimization in order to select the best hyper-parameters set for a specific model.

Genetic Algorithms (GAs) [25] are a metaheuristic optimization method, belonging to a broader class of evolutionary algorithms. GAs are inspired by the natural selection process, which mimics the evolution paradigm. The evolutionary training process starts from an initial set of solutions, or population $\mathscr{P}$, where every solution is given by a set of properties, called genes. A solution is also called an individual $h_{DGN} \in \mathscr{P}$. During training, an optimization objective is evaluated across populations, followed by the selection of the fittest solutions and their recombination using crossover and mutation operations. GAs are mainly used to solve global optimization objectives over problem domains with complex parameter spaces, demonstrating success in overcoming local minimums, when compared to gradient-based algorithms.

Algorithm 1 Deep Grid Net's training procedure in pseudocode.

procedure TRAIN($\mathscr{P}$)
  for $(\theta, h_{DGN}, t) \in \mathscr{P}$ **do**
    while `not end of training` **do**
      $\theta \leftarrow backprop(\theta | h_{DGN})$ ▷ backpropagation training for the current hyper-parameters in h_{DGN}
      $h_{DGN} \leftarrow eval(h_{DGN})$ ▷ evaluate the current model
      $h_{DGN} \leftarrow explore(h_{DGN}, \mathscr{P})$ ▷ select the new hyper-parameters set p
      `update` $\mathscr{P}$ `with new` $(h_{DGN}, \theta, t+1)$ ▷ update population $\mathscr{P}$
    end while
  end for
  return `top 5` h_{DGN} `individuals in` $\mathscr{P}$ ▷ returns the models with the highest scores
end procedure

GAs are used in our work for finding the optimal hyper-parameters set encoding h^*_{DGN}, that is, the optimal number of neurons in each layer, most suitable optimizer, the number and size of kernel filters used for the convolution operation and the best cost function for backpropagation. This allows us to determine the smallest neural networks structure,

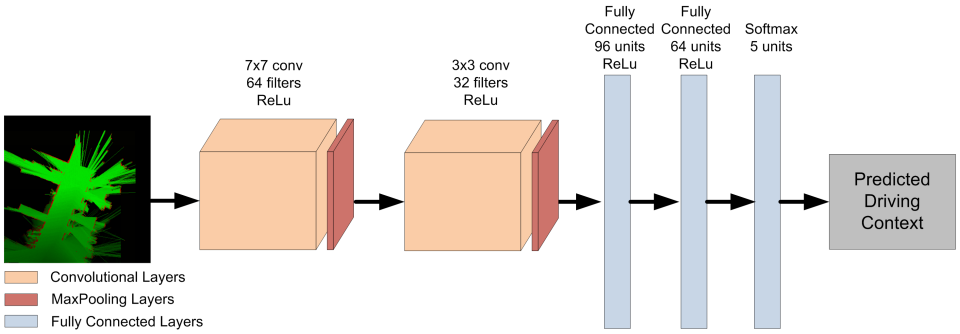

Fig. 4. DGN general architecture. It contains two convolutional layers, filtered by ReLu activation functions. Pooling and normalization operations are following each convolutional layer. The model contains three fully connected layers connected to a softmax activation function.

which can deliver accurate results, as well as real-time processing capabilities [26]. An *eval* function has been defined for evaluating the fitness function. Finding the optimal set of parameters is defined as:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, eval(\theta). \tag{14}$$

The proposed training method optimizes over a hyper-parameters solutions space, aiming at calculating the top individuals $h_{DGN} \in \mathscr{P}$ based on their fitness value, defined here as the accuracy of the deep neural network:

$$h^*_{DGN} = \underset{h_{DGN} \in \mathscr{P}}{\operatorname{argmax}}\, eval(backprop(\theta | h_{DGN})) \tag{15}$$

The optimal structure of the network is evaluated within the training loop for a given set of weights θ, DGN individual h_{DGN} and training step t. The weights θ are calculated using classical backpropagation, according to the maximum likelihood estimation defined in Eq. 8:

$$\theta \leftarrow backprop(\theta | h_{DGN}). \tag{16}$$

Once the training in Eq. 16 is completed, the hyper-parameters are evaluated based on h_{DGN} using the $eval(\cdot)$ fitness function. The new set of hyper-parameters are calculated by exploring the solution space with the help of the $explore(h_{DGN}, \mathscr{P})$ procedure. The training loop stops after 15 training epochs and returns the top 5 individuals, which have the highest fitness value. The pseudocode of our approach is given in Algorithm 1.

3.4. *DGN Architecture*

In the interest of constructing a grid representation of the driving environment we have defined a deep CNN topology which takes as input the OGs described in section 3.2. The

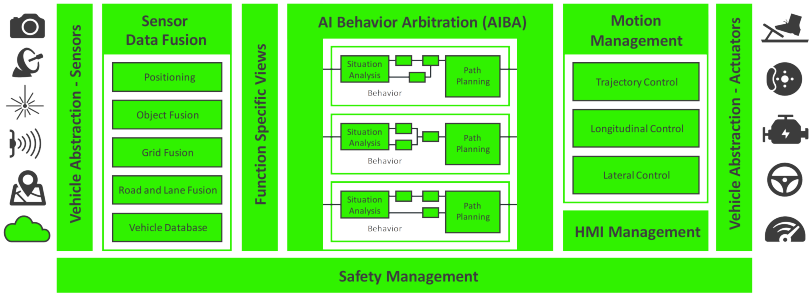

Fig. 5. Open EB robinos reference architecture (www.elektrobit.com).

resulted occupancy grids are exported as 128×128 images and fed as input into the CNN.

The neural network architecture is written in Keras [27], on top of the TensorFlow [28] library . Keras was chosen considering the ease in fast prototyping, as well as its capacity to run seamlessly on CPU and GPU.

The structure of the CNN, with a default hyperparameter set, is illustrated in Fig. 4. Reducing the problem than we want to solve to a multiclass classification task, we have chosen for performance evaluation competing neural networks like AlexNet [29], GoogLeNet [30], ResNet [31] and LeNET [32].

The *Deep Grid Net* architecture has been developed for deployment within EB robinos, where smaller activation maps are required in order to achieve real-time performance. The DGN's topology consists of two convolutional layers filtered by *Rectified Linear Unit* (ReLu) activation function. To find the number of kernel filters and their optimal size we provide intervals of values to the neuroevolutionary algorithm described in Section 3.3.

Each convolutional layer is followed by a pooling operation and a normalization layer (Batch Normalization). The network also contains three fully connected (FC) layers linked to a final Softmax activation function which calculates the driving context probabilities. In order to reduce the model overfitting, Dropout layers were added after the first two FC layers. The optimal number of neurons for the FC layers, as well as the loss function and the optimizer, are obtained based on the designed neuroevolutionary algorithm.

4. Experimental results

We describe in this section the experimental results obtained from the integration of DGN within EB robinos.

4.1. *EB robinos*

EB robinos is a functional software architecture from Elektrobit Automotive, with open interfaces and software modules, that manages the complexity of autonomous driving. The EB robinos reference architecture, illustrated in Fig. 5, integrates components following the sense, plan, act decomposition paradigm. Moreover, it also makes use of AI technology

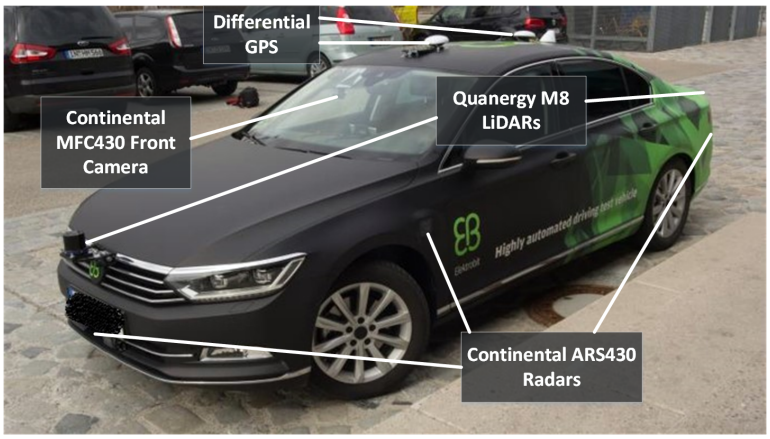

Fig. 6. Self driving test vehicle [33] used for real-world data acquisition.

within its software modules in order to cope with the highly unstructured real-world driving environment.

DGN is used within EB Robinos to select different driving strategies, depending on the context in which the ego-car has to drive autonomously.

4.2. *Training strategy*

The testing data contains occupancy grids collected from different driving scenarios. The data was collected on several road types in Germany, using the test vehicle (Volkswagen Passat) presented in Fig. 6 equipped with a front camera (Continental MFC430), a front and rear lidar (Quanergy M8) and front, rear and side radars (Continental ARS430).

The sensory data streams are fused into occupancy grids having a size of 125×125 and a resolution of $0.25m$. The recordings were performed during daytime, with mostly dry and sunny weather and include crowded, as well as light traffic conditions.

The data samples are saved during driving at time intervals ranging between 50 *ms* and 90 *ms* per cycle. Approximately 60.000 samples were obtained, containing different scenarios types: country roads, highways, inner city, parking lots, or traffic jam situations. From the total amount of samples, 65% were used for training, 25% for validation and 10% for testing. The validation data set provides an unbiased evaluation of a model fit on the training dataset while tuning model hyper-parameters. An important effort was allocated to select the proper validation data. Given the fact that the data is collected every 50-90 *ms* the consecutive samples can be very similar. To eliminate the problem of having very similar images in both training and validation datasets, the samples for validation were selected when slightly changes in the OGs structure are visually detected. The same procedure was used to create the dataset for testing.

Concerning the training process, a desktop computer was used, which was equipped with an NVIDIA GeForce GTX 1080 TI graphical processing unit (GPU).

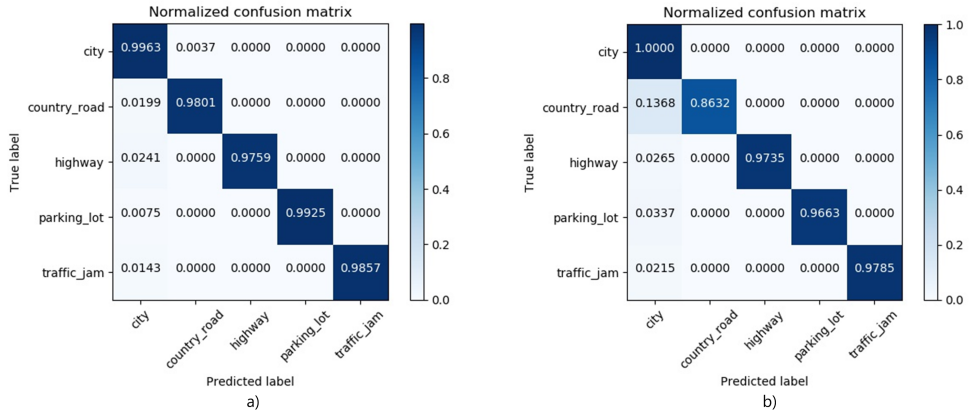

Fig. 7. Examples of confusion matrices generated during the neuroevolutionary training process. The figures are generated with different hyper-parameters from which it can be highlighted: a) mean-squared error as activation function and stochastic gradient descent (SGD) as optimizer; b) mean-squared error as activation function and adam as optimizer.

The classification model was trained from scratch, using the dataset described above and a learning rate α of 0.0001 for the backpropagation algorithm.

The structure of our network was determined based on the algorithm described in the Section 3.3. The scope was to search for the optimal optimizer, cost function, and number of neurons for the FC layers, together with the ideal number and size of the kernel filters used for the convolution operation. The following hyper-parameters were used as input for the neuroevolutionary training process:

- **Optimizers**: rmsprop, adam, Stochastic Gradient Descent (SGD), adagrad, adadelta, adamax, nadam;
- **Loss functions**: categorical crossentropy, mean squared error;
- **Number of neurons**: 16, 32, 64, 128, 258;
- **Number of kernel filters**: 16, 32, 64, 96;
- **Size of kernel filters**: 3, 5, 7, 9;

During the neuroevolutionary training process confusion matrices for each individual were generated. In Fig. 7 are presented examples of confusion matrices for two models containing the following hyper-parameters:

a) **optimizer**: SGD; **loss function**: mean squared error; **number of neurons**: 128 for both FC layers; **kernel filters**: $5 \times 5 \times 96$ and $5 \times 5 \times 16$ for the first and second convolutional layer, respectively
b) **optimizer**: SGD; **loss function**: mean squared error; **number of neurons**: 512 for the first FC layer and 128 for the second FC layer; **kernel filters**: $7 \times 7 \times 32$ and $3 \times 3 \times 16$ for the first and second convolutional layer, respectively

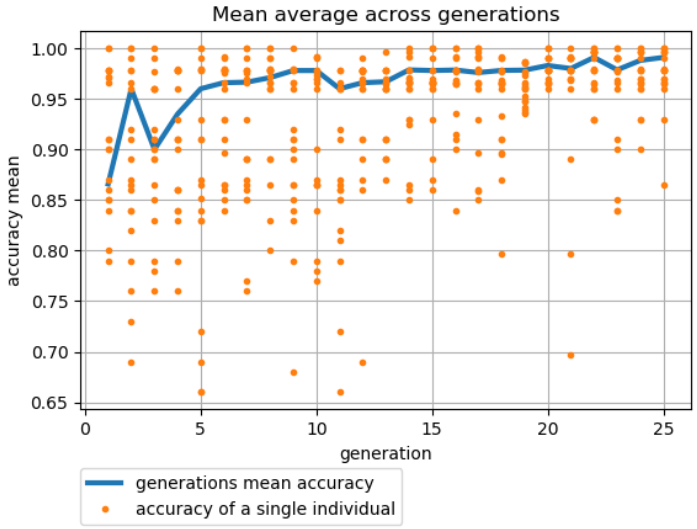

Fig. 8. Fitness function evolution during training.

It can be observed that the model evaluated in Fig. 7a) has a very good accuracy for all the classes, comparing with the model evaluated in Fig. 7b) where the accuracy is lower and the overfitting occurs. Calculating the confusion matrices for each individual is useful for evaluating the training process progress.

The fitness function evolution can be seen in Fig. 8. The GA evolved the DGN's hyper-parameters with a generation restraint value of 10, each generation consisting of 20 individuals. An individual represents a DGN neural network, with its structure described in Section 3.4 and with the hyper-parameters selected by the neuroevolutionary algorithm. An average classification accuracy is measured after each generation. When the last generation is reached, the individual with the best score is selected as h^*_{DGN}. Extending the hyper-parameters set used in our training method, we have reached an improved value of 99.1% fitness accuracy. The selected neural network architecture contains 96 and 64 neurons for the first and second fully connect layers, respectively and uses *nadam* as backpropagation optimizer and *categorical crossentropy* as loss function. The optimal number and size of the convolutional layers kernel filters were determined as following: $7 \times 7 \times 64$ filters for the first convolutional layer and $3 \times 3 \times 32$ filters for the second convolutional layer.

4.3. *Accuracy Evaluation*

The DGN's classification performance is summarized in the confusion matrix from Fig. 9a), where it can be observed that the classification accuracy is very high on the test dataset for every evaluated class. Minor differences in their per-class performance are visible. The classes *traffic jam* and *country road* present a higher detection accuracy due to a more distinctive structure of the occupancy grids.

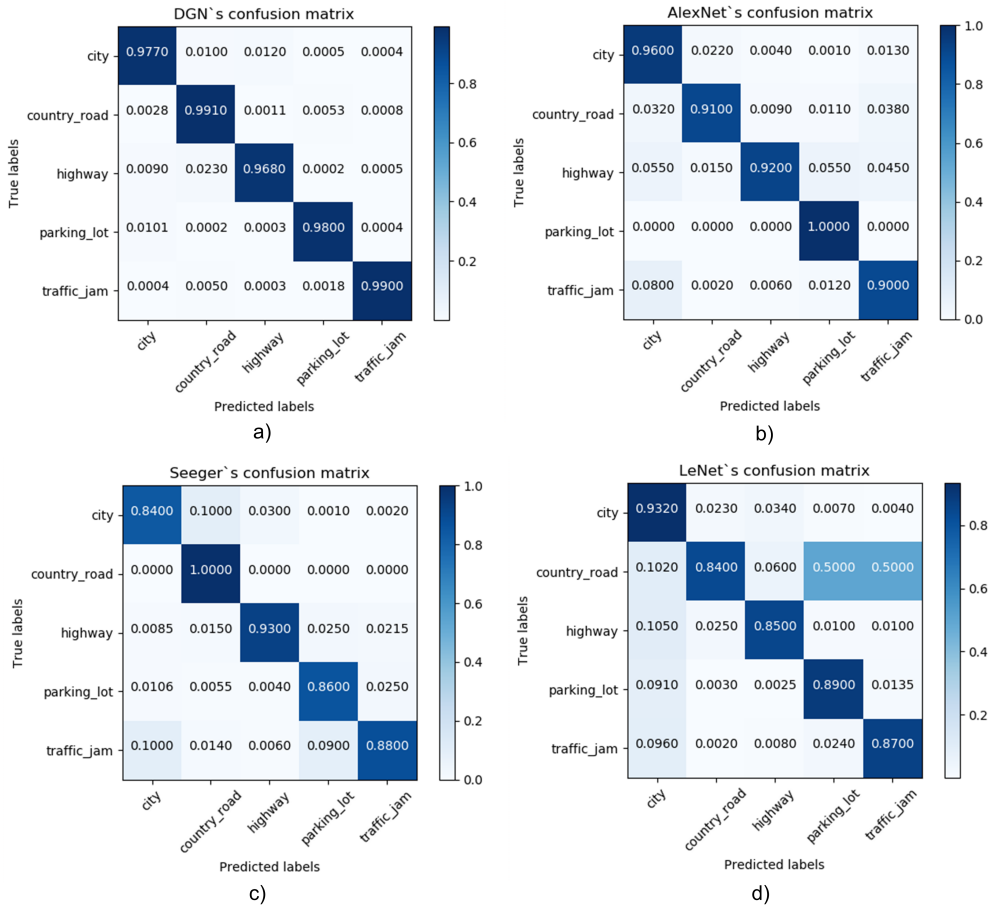

Fig. 9. Generated confusion matrices for performance evaluation. In this figure four confusion matrices are used to compare the accuracy of DGN, in figure a), with accuracy gained from the evaluation of: b) AlexNet, c) Seeger's method, d) LeNet.

A comparison of DGN's classification performance against state-of-the-art methods is presented in Table 1. The competitors are several network topologies, like AlexNet [29], or GoogleLeNet [30], as well as the algorithm from [17], which is the closest approach to our system. All algorithms were tested with respect to the same testing data, acquired with our test vehicle. The classification results obtained with DGN are clearly higher than the ones delivered by the competing estimators, this making DGN comparable with well known DNN topologies.

In order to have a better performance comparison overview, we have generated confusion matrices for the models which have more similarities with DGN. In Fig. 9b), c) and d) the confusion matrices for AlexNet, LeNet and Seeger's method [17] can be analyzed.

Interestingly enough, AlexNet and Seeger's models were able to provide a perfect score

Table 1. Comparison of driving context classification performance

Method	**Accuracy**	**Recall**	**Precision**	**F-measure**	**Specificity**
LeNET	0.88	0.91	0.93	0.92	0.86
GoogLeNet	0.94	0.96	0.97	0.97	0.97
ResNet	0.9	0.92	0.94	0.928	0.88
AlexNet	0.95	0.95	0.96	0.96	0.95
Seeger	0.91	0.9	0.92	0.92	0.93
DGN	**0.98**	**0,988**	**0,983**	**0,984**	**0.95**

Table 2. Test samples processing time comparison

	LeNet	GoogLeNet	ResNet	AlexNet	Seeger	**DGN**
Time (*milliseconds*)	175	930	820	710	620	**145**

accuracy for one of the classes, which can imply that our test set was not sufficient for these complex neural network architectures. Additionally, AlexNet provides a good accuracy for the class *city*, very close to DGN's performance. In accordance with the confusion matrix, Seeger's method has a lower classification accuracy comparing with DGN, even if the network complexity is higher. LeNet provides the lowest classification accuracy on our test dataset.

Adjacent to the high classification accuracy, DGN was designed to be used in real-time applications, like the EB robinos HAD platform. The speed of the algorithm was compared with the already mentioned competitors. In Table 2 the processing speed comparison is presented. In can be observed that DGN's is suitable for real-time application, having the best performances in terms of processing speed. LeNet model has comparable results. Nevertheless, the DGN's classification accuracy is much comparing with LeNet.

Comparing with the method defined in [17], DGN runs on single OG sample, without the need to accumulate grid data over time. DGN's architecture is simple, the number of layers and neurons being reduced to a necessary minimum, keeping in parallel an optimal accuracy.

5. Conclusion

In this paper, we have extended *Deep Grid Net* (DGN), a solution for driving context understanding, required by behavior arbitration components within Highly Autonomous Driving (HAD) systems. It has been designed to infer the driving context directly from Occupancy Grids (OGs), as opposed to traditional image based methods. We were able to show that a simplified convolutional neural network topology is sufficient to classify in real-time between different types of OGs, without the need of training large neural networks, such as AlexNet, or GoogLeNet.

One straightforward way to improve DGN's performance is to enlarge its training dataset, as well as increasing the number of OGs classes. New samples can be acquired especially through an automated process of synthetic training samples generation, as the GOL algorithm introduced by the authors in [6]. Candidates for new OGs classes can be construction sites, highway entrances or exits, lanes merging situations or accidents. Due to DGN's scalability, these additions can be directly implemented within the algorithm.

Another strategy to improve the DGN's is to fuse data from multiple sensors. In this direction images collected with the camera mounted on the test vehicle can be used as input into the deep learning algorithm, together with the OGs.

ACKNOWLEDGMENT

The authors would like to thank Elektrobit Automotive for the infrastructure and research support.

References

[1] L. A. Marina, B. Trasnea, T. T. Cocias, A. Vasilcoi, F. Moldoveanu and S. M. Grigorescu, Deep grid net (dgn): A deep learning system for real-time driving context understanding, *2019 Third IEEE International Conference on Robotic Computing (IRC)* 399–402 (2019).

[2] H. Zhu, K. Yuen, L. Mihaylova and H. Leung, Overview of environment perception for intelligent vehicles, *IEEE Transactions on Intelligent Transportation Systems* **18** 2584–2601 (Oct 2017).

[3] J. Janai, F. Güney, A. Behl and A. Geiger, Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art, *CoRR* **abs/1704.05519** (2017).

[4] A. Elfes, Using Occupancy Grids for Mobile Robot Perception and Navigation, *Computer* **22**(6) 46–57 (1989).

[5] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents))* (The MIT Press, 2005).

[6] S. Grigorescu, Generative One-Shot Learning (GOL): A Semi-Parametric Approach for One-Shot Learning in Autonomous Vision, in *Int. Conf. on Robotics and Automation ICRA 2018* (Brisbane, Australia, 2018, (to be published)).

[7] R. Goeddel and E. Olson, Learning semantic place labels from occupancy grids using CNNs, in *Intelligent Robots and Systems (IROS)* (IEEE/RSJ International Conference, 2016).

[8] P. Ondruska, J. Dequaire, D. Z. Wang and I. Posner, End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Network, in *Robotics: Science and Systems* (Workshop on Limits and Potentials of Deep Learning in Robotics, 2016).

[9] P. Ondruska and I. Posner, Deep tracking: Seeing beyond seeing using recurrent neural networks, in *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)* (Phoenix, Arizona USA, 2016).

[10] S. Hoermann, M. Bach and K. Dietmayer, Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling, in *arXiv:1705.08781* May 2017.

[11] D. Nuss, S. Reuter and M. Thom, A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-Time Application, in *arXiv:1605.02406* May 2016.

[12] D. Nuss, T. Yuan and G. Krehl, Fusion of laser and radar sensor data with a sequential monte carlo bayesian occupancy filter, *Intelligent Vehicles Symposium (IV)* 1074–1081 (2015).

[13] J. Dezert, J. Moras and B. Pannetier, Environment perception using grid occupancy estimation with belief functions, in *Information Fusion (Fusion)* (International Conference, 2015).

[14] F. Smarandache and J. Dezert, Modified PCR Rules of Combination with Degrees of Intersections, in *Proc. of Fusion* (USA, 2015).
[15] G. Shafer, A Mathematical Theory of Evidence (Princeton: Princeton University Press, 1976).
[16] S. Wirges, T. Fischer, J. B. Frias and C. Stiller, Object detection and classification in occupancy grid maps using deep convolutional networks, *CoRR* **abs/1805.08689** (2018).
[17] C. Seeger, A. Müller and L. Schwarz, Towards Road Type Classification with Occupancy Grids, in *Intelligent Vehicles Symposium - Workshop: DeepDriving - Learning Representations for Intelligent Vehicles, IEEE* (Gothenburg, Sweden, 2016).
[18] E. G. J. R. Jakub Safarik, Jakub Jalowiczor, Genetic algorithm for automatic tuning of neural network hyperparameters **10643**2018.
[19] Deep genetic network, *CoRR* **abs/1811.01845** (2018), Withdrawn.
[20] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley and J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, *arXiv preprint arXiv:1712.06567* (2017).
[21] T. Hinz, N. Navarro-Guerrero, S. Magg and S. Wermter, Speeding up the hyperparameter optimization of deep convolutional neural networks, *International Journal of Computational Intelligence and Applications* **17**(02) p. 1850008 (2018).
[22] F. Smarandache and J. Dezert, Advances and applications of DSmT for information fusion (Collected works), *American Research Press, USA* **1-4** (2004 - 2015).
[23] F. Smarandache and M. Alford, A Class of DSm Conditional Rules, in *COGIS 2009 International Conference* (Paris, 2009).
[24] A. Tchamova and J. Dezert, On the Behavior of Dempster's Rule of Combination and the Foundations of Dempster-Shafer Theory, in *Intelligent Systems (IS)* (6th IEEE International Conference, 2016).
[25] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd edn. (Springer Publishing Company, Incorporated, 2015).
[26] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando and K. Kavukcuoglu, Population based training of neural networks, *CoRR* **abs/1711.09846** (2017).
[27] F. Chollet *et al.*, Keras `https://keras.io`, (2015).
[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), Software available from tensorflow.org.
[29] A. Krizhevsky, I. Sutskevert and G. E. Hinton, Imagenet Classification with Deep Convolutional Neural Networks, in *Advances in Neural Information Processing Systems 25 (NIPS)* 2016.
[30] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *CoRR* **abs/1409.1556** (2014).
[31] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *CoRR* **abs/1512.03385** (2015).
[32] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, in *Proceedings of the IEEE* 1998, pp. 2278–2324.
[33] S. M. Grigorescu, B. Trasnea, L. Marina, A. Vasilcoi and T. T. Cocias, Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles, *CoRR* **abs/1906.10971** (2019).