A Multi-Platform Framework for Artificial Intelligence Engines in Automotive Systems

Liviu A. Marina
Department of Automation
Transilvania University of Brasov
Elektrobit Automotive
Brasov, Romania
marina.liviu.alexandru@unitbv.ro

Bogdan Trăsnea
Department of Automation
Transilvania University of Brasov
Elektrobit Automotive
Brasov, Romania
bogdan.trasnea@unitbv.ro

Sorin M. Grigorescu
Department of Automation
Transilvania University of Brasov
Elektrobit Automotive
Brasov, Romania
s.grigorescu@unitbv.ro

***Abstract*—It is becoming increasingly evident that data science and artificial intelligence are key technologies in the future of automotive industry. Artificial Intelligence (AI) successes are significant, with some limitations in terms of algorithms portability and code industrialization for mass production and deployment. This is because there are many domains where several software development process standards need to be applied.**

The approach presented in this paper tries to overcome this issue by creating a unique platform with multiple artificial intelligence engines. By using state-of-the-art AI libraries, together with target-independent software, tackles the challenge of driving context recognition in embedded systems, and uses it as a validation method.

***Index Terms*—artificial intelligence, deep learning, embedded AI, automotive, autonomous driving**

I. Introduction

Artificial intelligence, as an idea, firstly appeared soon after humans developed the electronic digital computing that make it possible. Just like any digital technology, AI has ridden waves of hype and gloom. Nowadays, AI is poised to unleash the next wave of digital disruption, as we are already seeing real-life benefits in several domains, such as computer vision, robotics, automotive and language processing.

Interest in AI has increased again lately, because of advances in fields such as deep learning (DL), underpinned by faster computers [1]. These machines are equipped with powerful graphics processing units (GPUs), which can process images between 40 and up to 80 times faster than a normal processor [2].

The analysis of large data volumes using specialized learning algorithms and pattern recognition provides the modelling of complex systems and dynamic processes. This enabled the automotive industry to focus more on AI research towards autonomous vehicles [3], with applications such as environment perception through occupancy grids, or synthetic data generation.

Although the successes of AI in the automotive domain are significant, it is worth remembering that it also has limitations [4]. For example, one major criticism of many AI systems is that they are often regarded as black boxes [5], which only map a relationship between input and output variables, based on a training dataset. A robust framework which also allows the visualization of intermediate stages in the training and deployment of AI models might be able to overcome these aspects.

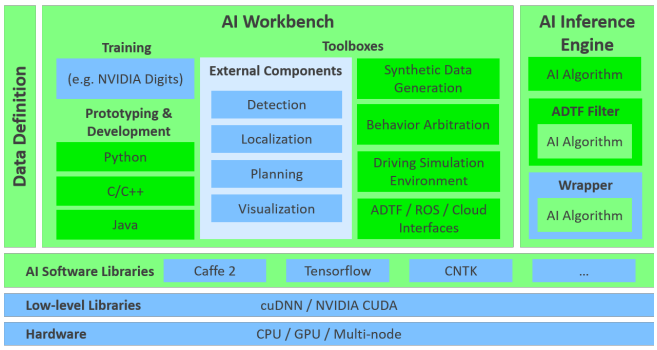


Fig. 1: AI Engines framework, following the two major operations from the deep learning workflow: training and inference, with additional layers for hardware integration and performance optimization

This article aims to contribute to the ongoing debate about the role of AI in the automotive industry, and to highlight the necessity of a robust framework for developing such systems. In particular, the contributions are:

(i) Providing an understanding of automotive DL applications such as occupancy grid classification [6], [7] and generative one-shot learning [8].
(ii) Highlighting the advantages of using platform-independent AI engines when designing, training and deploying DL systems in different environments.
(iii) Promoting the importance of complying DL design to software development process standards, such as ISO26262 [9] or ASPICE (Automotive Software Process Improvement and Capability Determination).

In this paper, the training and deployment of an optimized DL algorithm for indoor occupancy grid classification was performed, while the evaluation took place on a model car prototype.

The rest of the paper is organized as follows. In section II, a brief overview of the related work and state of the art advances is provided. In section III, the proposed AI framework is

described, together with the platform independent AI engines and the associated automotive use-cases. A short insight into code industrialization and compliance to process standards is also given in section III. Section IV is used for performance evaluation and provision of the experimental results, while section V is used to conclude the paper.

II. Related Work

One of the challenging problems of modern software development (and not only in AI industry) is that there are dozens of different frameworks doing literally the same things. Every big company that does some kind of machine learning has to have its own framework, together with the plethora of open source solutions.

When developing single AI applications it would be incredibly useful to use different frameworks [10], but merging them all takes a lot of development time and it distracts both data scientists and software developers from working on more important tasks. The solution can be a unique format of neural networks that can be obtained from any framework, and that can easily be deployed and used by developers and scientists.

The current state of the art in automotive AI development is quite distributed [11], [12], especially when discussing the available software libraries and taking into account the ratio between advantages and disadvantages. Predictions on autonomous vehicles implementations also highlight the importance of reduced complexity in development and deployment, in order to save costs and optimize the technology [13].

Deep learning techniques can be applied to many use cases in the automotive industry. There are several areas where the DL systems were significantly improved, like computer vision for lane or pedestrian detection, natural speech recognition or path planning. The AI open source software libraries used in this work are Tensorflow [14], Caffe2 [15] and CNTK [16]. These were chosen as the three best-performers out of the comparison considering several key characteristics.

The training of deep neural networks (DNNs) usually requires a large manually annotated database with input samples. Each software library uses its own input types, which means that a huge amount of training data is needed for having a modular framework. Collecting such a large input data set is not trivial, thus an alternative method is needed.

In order to try to bypass this manual annotation step, in our past work a semi-parametric approach to one-shot learning was used, coined GOL (Generative One-Shot Learning) [8]. Such an algorithm takes as input single one-shot objects, or generic patterns and templates, together with a small set of regularization samples used to drive the generative process, and generates as output new synthetic data.

GOL is intended to generalize on unseen data, while increasing the classification accuracy on synthetic data as well as possible. The training of GOL implies the learning of a set of optimal parameters Θ^* which maximize the generalization energies and classification accuracy.

III. Artificial Intelligence Framework

A. Overall architecture

The overall framework architecture, depicted in Fig. 1, follows the two major operations from the DL workflow: training and inference, together with the additional layers for hardware integration and performance improvement. The AI workbench is responsible for the training, prototyping and development of the system, by making use of existing methods and tools, and considering AI algorithms as toolboxes within the whole framework.

The first step consists on defining the requirements and collecting the necessary data for the training process. After the data is collected, it needs preprocessing, which includes annotation, normalization and filtering. Data preprocessing is a key step which enables proper prototyping of the AI algorithms, followed by a training and validation process. The training itself feeds the already defined data to the network, allowing it to learn a new capability, by reinforcing correct predictions and correcting the wrong ones.

Once the AI algorithm is tested and validated, it can be deployed as an AI Inference Engine. This represents the deployment of a trained model for each associated use-case, in order to evaluate new objects, and make predictions of real-world data with similar predictive accuracy.

B. Platform independent AI Engines

Deploying an AI algorithm inside an autonomous car is not trivial, due to the limitations of platform dependencies and decreased computation performance. Through our work a solution to overcome these limitations was proposed, by building platform-independent AI engines which are capable of various functionalities. Examples of such capabilities start from complex classification tasks and continue with scene understanding, objects perception and objects recognition tasks.

Our framework is capable, by using interfaces to the most important deep learning libraries, to integrate various neural networks topologies and to use data from different sensor types for evaluation. In this direction, a framework was developed, which contains inference engines for each of the chosen AI libraries: TensorFlow, Caffe2 and Cognitive Neural Toolkit (CNTK).

Caffe2 is the long-awaited successor to the original Caffe, with the main difference being that it is more scalable and light-weight.

TensorFlow presents the advantage that it supports more tasks than just DL, and it has an abstraction layer through computational graphs. It supports fine grain network layers that allow users to build new complex layer types without implementing them in a low-level language.

CNTK (Cognitive Toolkit) was initially developed for the advancement of speech recognition. CNTK supports both recurrent (RNN) and convolutional (CNN) types of neural models which make it a good candidate for handling image and speech recognition problems. It is faster than other frameworks, and it includes a thorough documentation.

Designed as platform independent, the AI engines can be easily used on desktop computers for research and development purposes and also can be integrated inside embedded platforms, as it will be demonstrated later in this paper.

In order to deploy our algorithm inside a stable and robust environment, our platform for AI engines was integrated inside Automotive Data and Time Triggered Framework (ADTF). This framework is described in detail in subsection IV-C.

Due to ADTF's limitations, in terms of supported programming languages and tool-set versions, the first step of our work was to link the selected AI libraries against ADTF software modules. To make the integration possible three C++ libraries' API were used to design wrappers for accessing the core functionalities. The wrappers were exported as dynamically linked libraries and referenced inside the implemented modules.

One of the ideas behind this approach was to have separated modules for each AI library. This will provide a big advantage when using deep learning features inside ADTF, and can help developers with less experience in the field to easily access and test the already existing algorithms. Another possibility is to use these modules for implementing new original alternatives for specific problems.

C. Occupancy grid classification

A possible use-case for which our AI framework can represent a solution is the understanding of the context in which an autonomous car is driving. Occupancy grids can be used to classify between various situations, with different classes representing the driving context, from which it can pointed out *city*, *motorway*, *highway*, *intersection*, *parking lot*, *roundabout* or *traffic jam* classes.

The occupancy grids can be constructed 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 [17].

The basic idea behind occupancy grids is the environment's division into 2D cells, each cell representing the probability, or belief, of occupation. Using the Dempster-Shafer rule, the occupancy maps can be built by taking into consideration the total degree of conflict between two sources of evidence:

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

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.

Exporting the occupancy grids as images will enhance the possibility of using a convolutional neural network to solve this classification problem. This use-case can be seen as a useful solution in autonomous driving, in order to solve issues linked to missing GPS signal in different locations.

In this paper the occupancy grid maps will be used to demonstrate the functionality of our AI framework, classifying the driving context in an indoor environment. A model car prototype is used as an embedded platform on which the algorithms can be deployed.

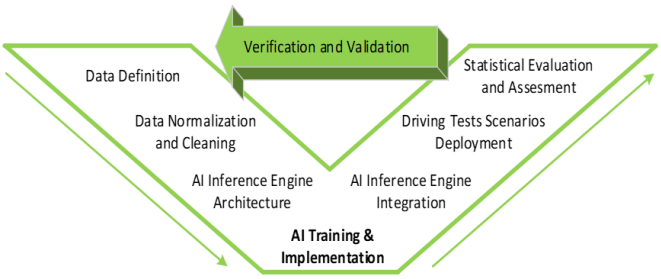


Fig. 2: V-model software development paradigm for process compliant prototyping and deployment

D. Compliance to process standards

In the automotive industry, one of the most important aspects in building software modules is the compliance to software development standards and its integration in vehicles embedded platforms. This ensures that the software is reliable and stable, such that it can be used in safety-critical applications.

It is important to approach automotive AI development from a more controlled V-model perspective, in order to address a lengthy list of challenges, such as requirements for the training, validation and test datasets, criteria for the data definition and preprocessing and impact of parameters tuning. In Fig. 2 a V-model is proposed for prototyping and development.

Data definition, normalization, and cleaning, together with its exploitation through the AI Inference Engine Architecture, are crucial development phases since the DNN's functional behaviour is the combined result of its architectural structure and its automatic adaptation through training. The integration of the obtained inference engines, associated with the deployment of the test scenarios and with statistical evaluation provide the required assessment tools for such an architecture model.

IV. Performance Evaluation

A. Dataset and Training

In order to demonstrate the applicability of our framework in autonomous driving related tasks, a deep learning algorithm was integrated inside an embedded platform to classify indoor driving scenarios.

Concerning the training and validation of our algorithm, a dataset was created using sensor inputs recorded with a fully equipped model car prototype, which can be seen in Fig. 4. The model car has been driven inside an office building and has recorded various data from the hallway and offices. A couple of samples are shown in Fig. 5.

The generated occupancy grids have been computed as 2D arrays, each covering 8 square meters in reality, and having a 25 cm resolution.

On one hand, 4000 samples were recorded, and annotated into two classes: *hallway* and *office*. On the other hand, the

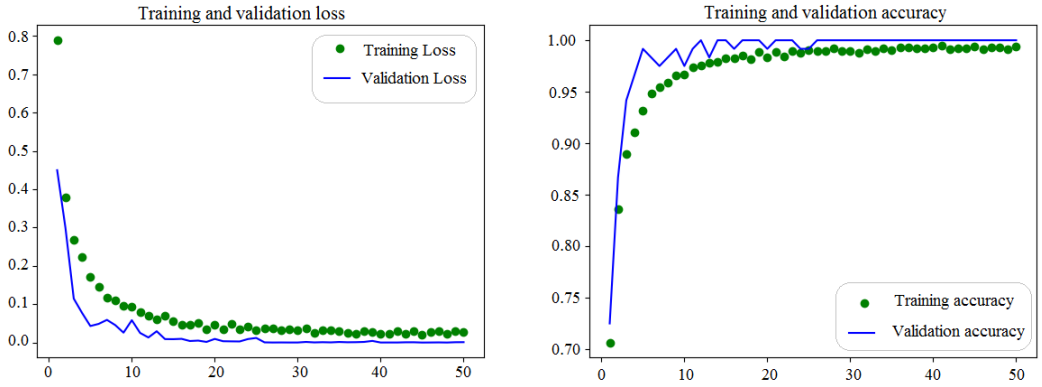


Fig. 3: Loss and accuracy progress over 50 training epochs. On the left side is the loss evolution, whereas on the right side is the progress of the overall accuracy, both on validation and training data sets.

same amount of samples were generated using GOL [8], to the extent of generalizing the output of the neural network, and to prevent over-fitting. From the entire dataset, 80% were used for training, 15% for validation and 5% for testing.

A user interface suitable for prototyping and training various deep learning topologies, and also capable of data preprocessing (i.e. annotation) was developed using Python programming language together with Keras API. One major advantage of the latter is that it can be used in both CPU and GPU configurations.

For the sake of reducing the time necessary for the training process, the following hardware set up was used: a desktop computer equipped with an Intel Core i7 7700K CPU, 32 GB RAM, and a high-performance graphics card, namely NVIDIA GeForce GTX 1080 Ti.

TensorFlow was used in order to create the neural network architecture, a model which was trained from scratch using the dataset described above. *Adam* [18], a method for stochastic optimization, was chosen as solver and categorical cross-entropy as loss function.

The criteria for this choice was the computation efficiency of the loss minimization, little memory requirements, and the applicability for problems with noisy and sparse gradients. *Adam* includes an adaptive moment estimation (m_t, v_t), having as update formulas:

$$(m_t)_i = \beta_1(m_{t-1})_i + (1 - \beta_1(\nabla L(W_t))_i \tag{2}$$

$$(v_t)_i = \beta_2(v_{t-1})_i + (1 - \beta_2(\nabla L(W_t))_i^2 \tag{3}$$

$$(W_{t+1})_i = (W_t)_i - \alpha \frac{\sqrt{1 - (\beta_2)_i^2}}{1 - (\beta_1)_i^2} \frac{(m_t)_i}{\sqrt{(v_t)_i} + \epsilon} \tag{4}$$

,where β_1, β_2 represent hyper-parameters and α represents the learning rate.

The deep learning algorithm used for indoor occupancy grids classification is represented by a deep convolutional neural network which constructs a grid representation of the indoor space. The model architecture was kept simple for faster computation without loosing the accuracy.

The neural network consists of two convolutional layers, with 32 and 64 Kernel filters, respectively. These are followed by a *Max-pooling* layer and two *Fully-connected* layers. *Rectified Linear Unit* (ReLu) filters each convolution, and a *Dropout* layer was included between the *Fully-connected* layers in order to reduce over-fitting. The model was trained over 50 epochs. The progress of loss and accuracy values can be observed in Fig. 3.

B. Model Car Prototype

In this section of the paper, a short description of the car prototype used for framework deployment will be provided. As it can be seen in Fig. 4, the prototype is 60 cm long, 30 cm wide and represents a model scale of 1:8. It features a wide range of close to serial-production-quality sensors and actuators. Additional features include a powerful processing unit, four-wheel drive and two steering axles.

The model car is equipped with ultrasonic proximity sensors, a 2D / 3D camera and a simple LIDAR distance sensor. Five low range ultrasonic sensors (model HC-SR04) are mounted on the front bumper and three on the rear bumper.

The camera (Asus Xtion Pro live) is located on the model car. It provides two image streams: one RGB image stream (VGA resolution) and one image which contains depth infor-

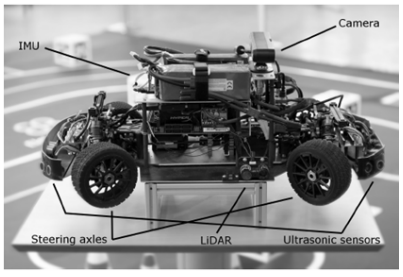
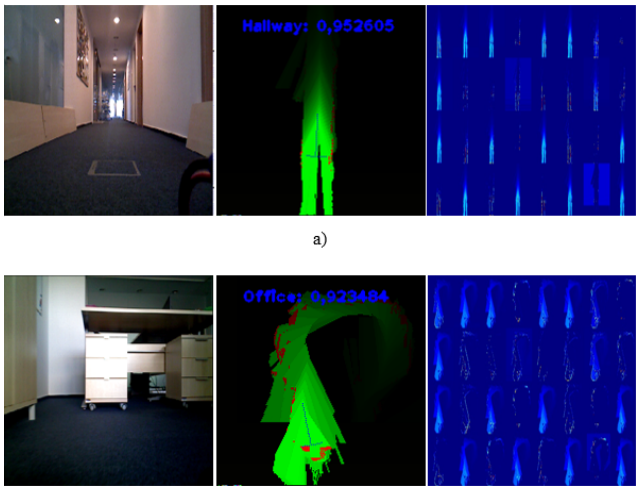


Fig. 4: Model car prototype used to deploy and test the AI engines inside an embedded environment.



a)

b)

Fig. 5: *Occupancy Grid* samples (middle) and the activations of model's first convolutional layer (right), along with their respective visual instances of the indoor environment (left).(a) Hallway. (b) Office Room.

mation. The latter is calculated from an infra-red grid that is projected on the ground.

In that manner, the distance to objects can be computed very precisely, with less than one centimetre deviation, even at a wide range (up to 3 m). This camera model cannot provide distance information closer than 60 cm.

A basic LIDAR sensor mounted on the right side of the car is used to measure the free space right next to the car. The ultrasonic sensors and the LIDAR are connected to Arduino Micro micro-controllers, which in their turn are linked to the central processing unit via USB.

Our work was based on the data provided by the depth camera, input used to map the indoor space into occupancy grids.

C. ADTF - Automotive Data and Time-Triggered Framework

Used to integrate and test our AI engines, ADTF is a platform which supports the development process in automotive software development. This environment has the advantage of a stable measurement framework, which is used in Advanced Driver Assistance Systems(ADAS) and can use almost any bus data (e.g. CAN, FlexRAY, Ethernet), as well as raw data from other sources.

From the main features of this framework, it can point out the stability for ADAS applications, the capabilities to integrate customers specific hardware, the real-time reproduction and visualization of the collected data during the recording.

There are some typical applications for ADTF framework, like measurement visualization, prototypical algorithm testing, validation of the measurement results, playback of the recordings as hardware-in-the-loop, and deployment of machine learning algorithms.

D. Indoor grid classification

The most important task of an autonomous vehicle is the localization inside an unknown environment. In the automotive industry, the localization is mainly realized using the GPS, a system which is not reliable for every decision an autonomous vehicle should take.

In this direction, the proposed use-case for our AI engines was the classification of the driving context using occupancy grids. In order to demonstrate that our framework can be used for this purpose, the versatility of our work was evaluated, by constructing and classifying indoor grids, using a Windows based desktop computer, a Unix based model car prototype and also Azure Cloud technology as deployment environments.

Examples of occupancy grids are visible in Fig. 5. The red cells represent the occupied space, while the free space is marked with green. The color intensity represents the degree of occupancy. If the intensity of the green color is higher, the probability of a cell to be free is also increased.

During this research, the occupancy grids were built using the depth camera mounted on the top of the model car and exported as images. Later the samples were used to train and validate the deep learning algorithm.

E. Experimental results

The portability of AI engines was demonstrated by deploying the DL algorithm inside a model car controlled by an embedded Unix operating system. In order to validate our work a deep neural network trained for indoor grid classification was used, to classify between *office room* and *hallway*. Visual samples which were collected during the test drive, together with associated occupancy grids and feature maps can be observed in Fig. 5.

Focusing on the framework's adaptability and robustness, a trivial algorithm was implemented, for a simple classification task. The results gathered from the model car were compared with the ones obtained by running our AI framework on a desktop computer and also inside a Azure cloud environment.

The comparison was made in terms of algorithm detection speed, the purpose being to optimize the neural network design to ensure reliable detection rate even in environments which doesn't benefit from GPU-grade computing power.

There are some major differences between the occupancy grids constructed for the hallway and office rooms, differences which help the algorithm to easily classify between the corresponding classes. To the extent of demonstrating the algorithm's accuracy, a confusion matrix was built, as it can be seen in Table I. Calculating the confusion rate needed 200 samples, which were tested against the ground truth. The result shows that the network topology can distinguish between *office room* and *hallway*, with a high accuracy.

Developing our software modules in compliance with standards required for automotive industry leads to the advantages of being platform independent. In this direction, our work was evaluated using our AI engines in different environments.

As it was presented above, the focus was to test our framework on an embedded platform. Concerning the pos-

TABLE I: Confusion matrix of the predicted indoor driving context

		Actual class	
		Office Room	Hallway
Pred. Class	Office Room	0.97	0.03
	Hallway	0.05	0 .95

sibility to run our software inside various environments, a desktop computer and a Cloud configuration were used for deployment. A CPU-based configuration was used to deploy the DL algorithm, in order to have the possibility to compare the detection rate performances. Using a desktop computer is effortless, once you already deployed the engines inside an embedded platform.

The code is cross-platform, so no changes are necessary. Due to the fact that the software modules were developed inside ADTF, the functionality is easily enabled by loading the modules. In terms of detection rate, there is no major difference between a desktop computer and the model car platform, as it can be observed in Table II.

F. Cloud Deployment

Another approach was to run the framework inside the Microsoft Azure cloud. The Azure Container Service is leveraged for starting multiple instances in parallel, allowing a high result throughput, as well as a quick validation of the classification components. Microsoft Azure was chosen for the availability of an unlimited number of machines configured with diverse resources.

To be able to test our AI engines in the cloud, the software modules were wrapped in ADTF and ran as part of a docker image. Each docker processes a list of recordings and saves the results in XML format. Using the cloud technology our algorithm performance was significantly improved, in terms of processing speed, as can be also seen in Table II.

TABLE II: Comparison of indoor environment classification performances

Platform	Frame Rate
Model Car	10 frames per second
Desktop Computer	12 frames per second
Cloud	25 frames per second

V. Conclusions

In this paper, we have designed a framework which can be used as a starting point for AI code compliance. We believe that our work's main contribution is the AI framework's usage for development and deployment, not only as prototype, but also for series production.

We have demonstrated the possibility of designing and implementing a complete framework, capable to cover all algorithm phases, from training to deployment. The possibility to use various deep learning libraries inside a single framework is very useful in terms of production efficiency.

Comparing with the already existing frameworks, which cover limited technologies and specific neural networks topologies, our work can be used for different tasks and on different platforms. We have proven the possibility of running our AI engines in three different environments, a desktop computer, an embedded platform, and into Microsoft Azure Cloud, with minimum configuration changes.

One way of improving our framework is extending it for more automotive use-cases, like behaviour arbitration, path planning or environment perception. Another possibility is to deploy the framework inside a real car, fully equipped for driving in real-world traffic scenarios.

ACKNOWLEDGMENT

We hereby acknowledge Elektrobit Automotive for providing the infrastructure and for support during research.

References

[1] T. Hwang, "Computational power and the social impact of artificial intelligence," *CoRR*, vol. abs/1803.08971, 2018.
[2] J. Jaros and P. Pospichal, "A fair comparison of modern cpus and gpus running the genetic algorithm under the knapsack benchmark," vol. 7248, pp. 426–435, 04 2012.
[3] M. Hofmann, F. Neukart, and T. Bäck, "Artificial intelligence and data science in the automotive industry," *CoRR*, vol. abs/1709.01989, 2017.
[4] M. Jamilly, F. Nagle, and C. Ross, "Limitations of ai," *ITNOW*, vol. 60, no. 1, pp. 16–17, 2018.
[5] D. Gunning, "Explainable artificial intelligence (xai)," in *Technical report, DARPA/I20*, 2016.
[6] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," *arXiv preprint arXiv:1705.08781*, 2017.
[7] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," *arXiv preprint arXiv:1602.00991*, 2016.
[8] 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, 21-25 May 2018.
[9] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, and R. Palin, "Safety cases and their role in iso 26262 functional safety assessment," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2013, pp. 154–165.
[10] R. Fonnegra, B. Blair, and G. Daz, "Performance comparison of deep learning frameworks in image classification problems using convolutional and recurrent networks," pp. 1–6, 08 2017.
[11] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," *CoRR*, vol. abs/1705.00346, 2017.
[12] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving," vol. abs/1504.01716, 2015.
[13] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute, 2017.
[14] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.
[15] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2014, pp. 675–678.
[16] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang *et al.*, "An introduction to computational networks and the computational network toolkit," *Microsoft Technical Report MSR-TR-2014–112*, 2014.
[17] G. Shafer, "A mathematical theory of evidence," 1976.
[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." *CoRR*, vol. abs/1412.6980, 2014.