Deep Learning Metadata Fusion for Traffic Light to Lane Assignment

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

"Doctor rerum naturalium"

der Georg-August-Universität Göttingen

Im Promotionsprogramm Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von
Tristan Matthias Langenberg
aus Düsseldorf

Göttingen, 2019

Betreuungsausschuss:

Prof. Dr. Florentin Wörgötter, III. Physikalisches Institut, Universität Göttingen
Dr. Minija Tamosiunaite, III. Physikalisches Institut, Universität Göttingen
Dr. Thomas Hörnlein, Mercedes-Benz Cars Entwicklung und Forschung, Daimler AG

Mitglieder der Prüfungskommission:

Referent: Prof. Dr. Florentin Wörgötter, III. Phys. Institut, Universität Göttingen
Korreferent: Prof. Dr. Carsten Damm, Institut für Informatik, Universität Göttingen

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Jens Grabowski, Institut für Informatik, Universität Göttingen
Prof. Dr. Stephan Waack, Institut für Informatik, Universität Göttingen
Prof. Dr. Wolfgang May, Institut für Informatik, Universität Göttingen
Prof. Dr. Minija Tamosiunaite, Institut für Informatik, Universität Kaunas, Litauen

Tag der mündlichen Prüfung:

Freitag, den 26. Juli 2019

Kurzfassung

Der Schwerpunkt dieser Dissertation ist eine neuartige tiefgreifende Fusionsmethode zwischen heterogenen Metadaten und Bilddaten zur Lösung des Zuordnungsproblems von Ampeln zu Fahrspuren. Die Ampel zu Fahrspurzuordnung gehört dem Forschungsbereich der autonomen Robotik bzw. des autonomen Fahrens an und wird unter Verwendung von Methoden der künstlichen Intelligenz bearbeitet.

Die Arbeit verwendet einen Datensatz mit über 45.000 Einzelbildern aus 848 komplexen Straßenkreuzungsszenarien in Deutschland. Als Basis besitzt jedes Kreuzungsszenario die Verbindung zwischen Ampeln und Fahrspuren als Referenzinformation und ist mit den folgenden Metadaten annotiert: Ampeln, Fahrspurmarkierungen, Fahrspurrichtungspfeilen und Fahrspurschildern. Es wird eine optimierte Vogelperspektiventransformationsmethode vorgestellt, die unabhängig von extrinsischen Kameraparametern ist und ein vollständiges Vogelperspektivenpanorama aus zusammengesetzten Einzelbildern erzeugt. Diese Methode wird für die Bildvorverarbeitung eingesetzt und ermöglicht eine effiziente Annotation von Fahrspurmarkierungen in der Vogelperspektive.

Zuerst wird gezeigt, dass ein neuronales Faltungsnetz ein Zuordnungsproblem in ein Regressionsproblem transferieren kann, um alle relevanten Ampeln ihren zugehörigen Fahrspuren zuzuweisen. Es wird ein Positionsvektor als Faltungsnetzausgabe definiert. Dieser kodiert alle relevanten Ampelpositionen als binäre Information. Dadurch wird die Ampel zu Fahrspurzuordnung ausschließlich visuell gelöst. Des Weiteren wird der visuelle Ansatz um einen tiefgreifenden Metadatenfusionsansatz erweitert. Dieser Fusionsansatz ermöglicht heterogene Metadaten in ein neuronales Faltungsnetz zu fusionieren. Dabei werden die Metadaten in verschiedene Metadatenmerkmalkarten transformiert. Diese Metadatenmerkmalkarten werden mittels einer elementweisen Multiplikation und einer adaptiven Gewichtungstechnik mit dem globalen Mittelwert der ausgewählten Fusionsebene im neuronalen Faltungsnetze fusioniert. Der tiefgreifende Metadatenfusionsansatz wird auf alle Arbeitsschritte geprüft, gegen regelbasierte, Metadaten getriebene und visuelle Ansätze verglichen und um ein Sequenzmodel erweitert. Weiterhin wird ein professioneller Probandentest durchgeführt, um die menschliche Leistung für dieses Zuordnungsproblem zu messen und als Maßstab zu definieren.

Im Ergebnis erreicht der tiefgreifende Metadatenfusionsansatz eine mittlere Genauigkeit von 93,7 % und übertrifft signifikant regelbasierte, metadatenbasierte und bildbasierte Ansätze. Er übertrifft auch die gemessene menschliche Leistung für den vollständigen Datensatz in der Genauigkeitsmetrik (+2,7 %) und dem F_1wert[1] (+4,1 %). Jedoch

[1]Der F_1wert setzt sich aus der Richtig-Positiv-Rate (engl. Precision) und der Sensitivität (engl. Recall) zu gleichen Teilen zusammen.

erzielen die gemessene menschliche Leistung und der tiefe Metadatenfusionsansatz eine nahezu identische Richtig-Positive Rate von 92,9 ±1,3 %. Außerdem wird festgestellt, dass eine frühe Fusion der Metadaten am effektivsten ist und alle fusionierten Metadatenmerkmalkarten sich positiv auf die Ergebnisse auswirken. Ferner hat sich gezeigt, dass der beste Fusionsoperator die elementweise Multiplikation ist. In Anlehnung an die menschliche Wahrnehmung ist festzustellen, dass sich die Ergebnisse mit sinkendem Abstand zur Haltelinie steigern.

Stichwörter neuronale Faltungsnetze, tiefgreifende Fusion, intelligente Transportsysteme, Robotik und Automatisierung, Ampelassistenz.

Abstract

This dissertation focuses on a novel deep fusion method with heterogeneous metadata and image data to resolve the one-to-many traffic light to lane assignment problem. The traffic light to lane assignment belongs to the research field of autonomous robotics or driving and is handled using artificial intelligence.

The work uses a dataset with over 45 thousand frames from 848 complex intersection scenarios in Germany. Each intersection scenario has as a ground truth, the traffic light to lane connections and is annotated with the following metadata: traffic lights, lane line markings, lane arrow markings, and lane signs. An optimised inverse perspective mapping method is introduced which is independent from extrinsic camera parameters and creates a stitched inverse perspective mapping full panorama image. This method is employed for image data preparation and enables an efficient annotation of inverse perspective mapping lane line markings.

At first, it is shown that a convolutional neuronal network can transfer an assignment problem in a regression problem to assign all relevant traffic lights to their associated lanes. Here, an indication vector defines the output of the network. The vector encodes all relevant traffic light column positions as binary information. This introduced strategy resolves the traffic light to lane assignment problem by vision, exclusively. Furthermore, the vision solution is enhanced by a deep metadata fusion approach. This approach is able to fuse heterogeneous metadata into a convolutional neural network. It transforms the metadata into several metadata feature maps. These metadata feature maps are fused into the convolutional neural network by means of an element-wise multiplication and an adaptive weighting technique with the global average of the selected fusion layer. The approach is examined for all working steps, compared against rule-based, only-metadata, and only-vision approaches and extended by a sequence approach. To appraise the deep metadata fusion approach in an expert manner, a subjective test is conducted that measures the real human performance for the traffic light to lane assignment and defines an independent baseline.

As result, the deep metadata fusion approach reaches a mean accuracy of 93.7 % and outperforms rule-based, only-metadata, and only-vision approaches significantly. It also outperforms the human performance in the accuracy (+2.7 %) and F_1score[2] (+4.1 %) metric for the full dataset. However, the human performance and deep metadata fusion approach achieve an almost identical mean precision result with 92.9 $\pm$1.3 %. Additionally, it results that an early fusion is most effective and all fused metadata feature maps have a positive effect on the results. The ideal fusion operator is the

[2]The F_1score is a composted and equally weighted metric of the precision and recall metrics.

element-wise multiplication and the results increase the closer the vehicle approaches the stop line similar to humans perception.

Keywords Convolutional Neural Networks, Deep Fusion, Intelligent Transportation Systems, Robotics and Automation, Traffic Light Assistance.

Contents

List of Figures

List of Tables

1 Introduction

Scope Self-driving cars will be the next big technology step for our society in the coming decade. At the present time, many car manufacturers spend a great deal of effort in research of autonomous vehicles. Here, the goal is to reach autonomous driving without any pedals or a steering wheel, also known as level five driving [1], to absolve the driver from his legal responsibility. This technology also influences the development of level three (highly automated driving[3]) and level four (fully automated driving[4]) vehicles. According to the statistic, level three and level four will be in the European market by car manufacturers with a large quantity in 2020 and 2025 [2].

In addition to the development of the technology in the car, a key component is the development of infrastructure technology in form of high definition maps (HD maps). HD maps support sensor-based functions of self-driving cars by use of their detailed knowledge about the environment. They are needed for autonomous driving in order to enable the self-driving car to make decisions in complex driving situations, localise itself with high precision in relation to its surrounding world, and to drive in accordance with the needs of traffic participants [3]. Another key component are outstanding government regulations, which have to be enacted to clear the liability in case of accidents. Otherwise, car manufacturers are not allowed to sell self-driving cars in large quantities.

Furthermore, hardware manufacturers such as NVIDIA and Intel are developing their own autonomous driving kits [4], [5]. These kits include efficient hardware performance and custom-built software algorithms for autonomous driving systems. At the moment, almost every car manufacturer or supplier cooperates with one of these hardware manufacturers since a powerful and resource efficient hardware platform is necessary to develop machine learning or deep learning algorithms. Deep learning is considered as an enabler for autonomous level five or lesser driving and replaces gradually conventional image processing algorithms, e.g. specifically Hough transformation for cycle detection in traffic lights or histograms of orientated gradients for general object classification. The key for deep learning is exactly annotated data, which has to be recorded and

[3]Highly automated driving: The vehicle drives in defined environments without any driver input. In unknown situations, the driver has to take over control.

[4]Fully automated driving: The vehicle drives without any driver input. In unknown situations, the vehicle is able to stop safety at a parking place.

annotated by humans as well as a large amount of data also for rare and challenging driving situations.

Topic This work relates to Advanced Driver Assistance Systems (ADAS). ADAS systems form the basis for self-driving cars and are used from level two (partly automated driving[5]) upwards. It is part of the research field of intelligent transportation systems and autonomous robotics. In detail, this dissertation deals with the topic of "deep learning metadata fusion for traffic light to lane assignment". The aim is to develop a reliable Traffic Light to Lane Assignment (TL2LA) function for Traffic Light Assistance (TLA) systems by use of a Convolutional Neural Network (CNN). This resolves the special issue in which the traffic light is relevant for which lane at signalised intersection scenarios. For this purpose, a novel deep metadata fusion approach is developed. This approach creates the TL2LA function by fusing image data and additional metadata within a CNN.

Mathematically speaking, the TL2LA function could be expressed by a many-to-many assignment problem. The problem will reduce to a one-to-many assignment problem if the ego-vehicle lane only is considered. An assignment problem would be resolved analytically with an optimisation of the Kuhn-Munkres algorithm[6] if the problem was transformed to the graph theory and all connection properties were known in form of a performance matrix, cf. [6]. In case of an analytic search algorithm solution, the complexity of the TL2LA problem can be expressed by $O(n^2)$ under the condition that n indicates the number of traffic lights T and lanes L. However, an analytical solution is very difficult because of the unknown performance matrix properties. The assignments, e.g. in figure 1.1, can be expressed by an allocation matrix

$$M_{alloc} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}}_{T \text{ traffic lights}} \left.\vphantom{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}\right\} L \text{ lanes}, \tag{1.1}$$

which contains the assignment of all traffic lights T to all lanes L. In this matrix, a one represents a relevant traffic light for the respective lane and a zero indicates a non-relevant connection. In total, there are 2^n theoretical solutions. Figure 1.1 underlines

[5]Partly automated driving: The vehicle drives for a short time without any driver input. The driver has to permanently monitor and take over control of the vehicle at any time.

[6]Also known as the Hungarian method with an initial complexity of $O(n^4)$.

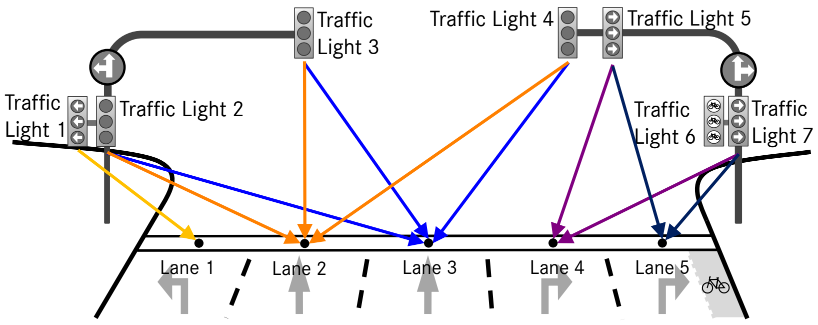


Fig. 1.1 Exemplary presentation of the traffic light to lane assignment (TL2LA) function. Traffic light 1 and traffic light 2 are positioned close together but are not relevant for the same lanes. Traffic light 4 is mounted above lane 4 but is relevant for lane 2 and lane 3. Traffic light 6 is a non-relevant traffic light for all lanes. This figure is taken from figure 4.1 in [7].

the challenges of this special issue[7]: Traffic light 1 and traffic light 2 are positioned close together but are not relevant for the same lanes. Traffic light 4 is mounted above lane 4 but is relevant for lane 2 and lane 3. Traffic light 6 is a non-relevant traffic light for all lanes and must be ignored.

Many drivers sometimes have trouble to assign the relevant traffic lights to their current lane. Causes for this could be distraction at the wheel or unknown urban environments. If this happens, most drivers will decelerate to gain more time for the decision before the end of the intersection scenario is reached as marked by the stop line. The TL2LA function will help to avoid accidents and save lives in the future. Today, about 4.0 % of all (≈2,600) fatalities per year are caused by motorcar accidents with a red light offence in Germany. And about 2.3 % of all (≈300,000) registered red light offences per year cause accidents with personal injury in Germany [8], [9]. Furthermore, 800 (1.9 % of ≈42,000) fatalities were registered on average each year in two-vehicle crashes that involved at least one driver who ran a red light in the USA from 1997 to 2004 [10]. This makes the TL2LA function a necessary aspect for all following TLA systems in the field of ADAS and self-driving cars. Some conceivable TLA systems are listed below:

- Signalling of the current traffic light state (information)
- Automatic warning of red traffic lights (warning)
- Emergency braking on stop lines (action)

The first TLA system on the market is the Traffic Light Information System from Audi

[7]The generic intersection scenario, see figure 1.1, would have $2^{(T+L)} = 4096$ assignment possibilities without logical restrictions.

[11]. This system uses HD maps for the traffic light position, state and relevant information. It acquires its traffic light information via Interface to Vehicle communication (I2V) from local traffic centres. That is why it is only available in ten U.S. cities.

The approach is to develop an independent TL2LA function which makes the assignment within the first attempt without any prior knowledge about the current intersection scenario. This excludes the usage of cloud management data, e.g. statistical observations of the current intersection scenario or any I2V. Thus, the approach can be used in every urban environment without the need of HD map information about relevant traffic lights. Rather, the idea can be used to validate or even generate this information for HD maps.

Database This thesis uses as a database the DriveU traffic light dataset [12]. In comparison to other public traffic light datasets (LaRA[8] [13], VIVA[9] [14], and Bosch Small[10] [15]), it has the most annotated traffic lights and states with entire intersection scenario frame sequences. In addition to the already annotated traffic lights in the DriveU dataset, the images were annotated by humans with lane line markings, lane arrow markings, and lane signs, as well as with connections between traffic lights and lanes. This connection between traffic lights and lanes represents the TL2LA ground truth.

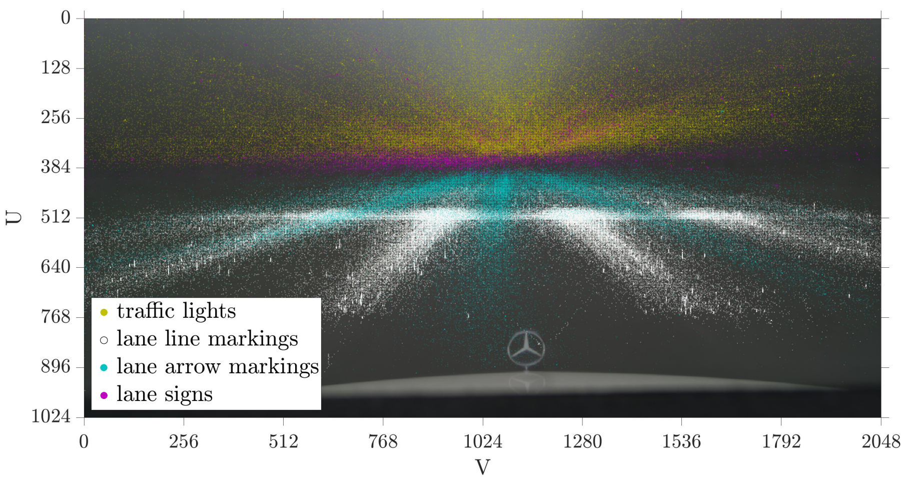


Fig. 1.2 Distribution of all annotated metadata objects in the left camera image (traffic lights, lane line markings, lane arrow markings, and lane signs). The metadata objects are symbolised as coloured markers centred on their image position. The background of the figure forms the mean left camera image of all annotated camera images.

[8]La Route Automatise: http://www.lara.prd.fr/benchmarks/trafficlightsrecognition.
[9]Vision for Intelligent Vehicles and Applications: http://cvrr.ucsd.edu/vivachallenge/index.php/traffic-light/traffic-light-detection/.
[10]Small Bosch Traffic Light Dataset: https://hci.iwr.uni-heidelberg.de/node/6132.

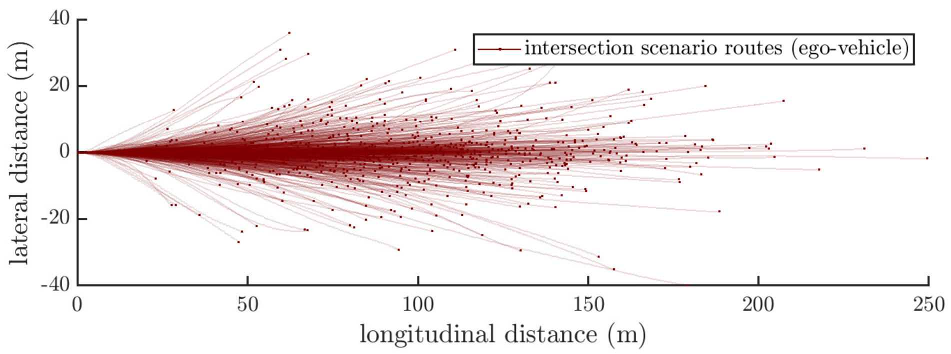


Fig. 1.3 Visualising of all intersection scenario routes driven by the ego-vehicle and standardised on the same starting point. Each line represents one of the 848 database sequences and the total travelled distance is 78.9 km.

Figure 1.2 shows a distribution of all annotated metadata objects[11]. Each metadata object (traffic light, lane line marking, lane arrow marking, or lane sign) is visualised as a round and coloured marker at its centred image position. The background fills the mean left camera image of all annotated camera images. Moreover, an annotated example image is available in appendix section 7.1.

The database was recorded using a stereo camera. Each camera image has a resolution of (1024, 2048, 3) RGB pixels. The frame rate is 15 fps and every third frame is annotated. In sum, there are 45,317 frames annotated. The amount of uncompressed data is about 1.6 TBytes[12]. The total travelled distance of all intersection scenarios is 78.9 km. The intersection scenario routes of all 848 sequences are visualised conceptually in figure 1.3. Each ego-vehicle route is plotted for the lateral and longitudinal distance and is standardises on the same starting point at zero. The entire travelled distance was driven at an average ego-vehicle speed of 31.3 $\frac{\text{km}}{\text{h}}$. In the course of this drive, the average number of lane changes per intersection scenario was 0.28 lane changes and the average number of traffic light state changes per intersection scenario, e.g. from green to yellow colour, was 2.3 traffic light state changes. Furthermore, the average number of lanes per intersection scenario is 3.3 lanes and the average number of traffic lights per intersection scenario is 6.7 traffic lights in the database.

As described, the approach uses already detected and classified traffic lights for the TL2LA function. Consequently, traffic light recognition (TLR) is not part of this thesis because this task is already well addressed in literature, cf. [15]–[23]. Furthermore, the recognition of the additional annotated lane line markings, lane arrow markings,

[11]The database acquisition, data annotation, and preparation of a data annotation tool was also part of this work and preceded the actual development of the TL2LA function.

[12]1.6 TBytes (image data) = 1024 (pixels) · 2048 (pixels) · 3 (RGB channels) · 45317 · 3 (all images) · 2 (left and right stereo camera) · 1 bytes per pixel.

and lane signs is also not part of this thesis and is already addressed by other reference literature, cf. [24]–[27].

Outline The thesis is structured in the following parts. The first part contains necessary background knowledge about the applied methods in the two quoted manuscripts. The functionality of neural networks is explained and the most important layer types and hyper parameters are addressed to set up a CNN. In particular, the part explains the normalisation and merging of convolutional feature maps that are necessary for the deep metadata fusion approach. Furthermore, a separate Inverse Perspective Mapping (IPM) method is introduced in order to transform the road part of the camera image into a bird's-eye view. This is used for the pre-processing of input images. The IPM method is also used to annotate lane line markings, which are required as metadata for the deep metadata fusion approach. The second part is a manuscript of a published paper at the IEEE Intelligent Transportation System Conference (IEEE ITS[13]). The paper explains a CNN model approach to develop a TL2LA function. The assignment problem is converted into a regression problem with the help of an output regression vector, which is resolved by a CNN. The third part is a second manuscript of a published IEEE paper in the Robotics and Automation Letter (IEEE RA-L[14]). This paper describes in detail the novel deep metadata fusion approach to create a reliable TL2LA function by combining image data with additional metadata within a CNN. It is based on the IEEE ITS CNN model approach and uses its findings, e.g. for drawing a comparison to other approaches and for the design of another subjective test. Both manuscripts have a prefixed exposition to underline their contribution in the context of this thesis and both are followed by an additional section to contribute complementary investigations. The last two parts deal with a discussion about the manuscripts as well as a summary and outlook of my dissertation.

[13]The authors are T. Langenberg and F. Wörgötter. Contribution of T. Langenberg (first author): implementation and evaluation of experiments and writing of the paper. Contribution of F. Wörgötter (second author): scientific advice and paper corrections.

[14]The authors are T. Langenberg, T. Lüddecke, and F. Wörgötter. Contribution of T. Langenberg (first author): implementation and evaluation of experiments and writing of the paper. Contribution of T. Lüddecke (second author): review of the approach and paper corrections. Contribution of F. Wörgötter (third author): scientific advice and paper corrections.

2 Background Knowledge of Applied Methods

2.1 Theory of Convolutional Neural Networks

The concept of neural networks denotes an interdisciplinary research field of system theory, models and methods regarding biological neurons, and neuron associations [28]. The motivation for the usage of neural networks results from the functionality and performance of their biological example. The idea is to adapt the neural system from animals, especially from humans, and model this into a usable algorithm.

In connection with neural networks, Convolutional Neural Networks (CNN) have become more famous for several image classification tasks in the last few years. CNNs are the state-of-the-art for image processing tasks since the victory of the AlexNet CNN [29] at the ImageNet Large Scale Visual Recognition Challenge[15] (ILSCRC) in 2012. Since then, CNNs are built with more and more convolutional and other special layers. Some of the most popular CNNs are the VGG-16 [30] from 2014, the GoogLeNet [31] with their interception module, and Microsoft's ResNet [32] with 152 neural network layers from 2015. All of these CNNs have won the past ILSCRC challenges.

In addition to the already mentioned image classification task, CNNs are also used for object detection, image segmentation and regression tasks. These tasks are always based on the fundamental property of pattern recognition, which is the main characteristic of CNNs.

Moreover, pre-trained CNNs are used for relative image processing tasks in practical applications, e.g. the above-mentioned VGG-16 CNN or GoogLeNet are often used as pre-trained CNNs for image classification. On the one hand, transfer learning is possible. All convolutional layers and their weights are copied from a pre-trained CNN to a new CNN. Only the remaining fully connected layers are trained while the convolutional layers remain static. This saves training run-time and hardware resources. On the other hand, adaptive learning exists. The complete CNN topology and its weights are copied from a pre-trained CNN to a new CNN. The new CNN is trained for another task

[15] The ILSCRC is a yearly image classification competition.

with the copied weights, which were used for initialisation. This also reduces training run-time and sometimes achieves better test results.

2.1.1 Definition of Deep Learning

The term deep learning is often mentioned in the context of CNNs but is not limited to image processing tasks only. It describes a special realisation of Artificial Neural Networks (ANN) with CNNs, which themselves belong to the class of machine learning algorithms [33]. Deep learning is defined as an approach that solves problems using representation learning. Computers are enabled to learn complex concepts based on simpler concepts by representations that are expressed in terms of other simpler representations [34, p. 5]. Deep learning is realised with a hierarchical construction of many neural network layers, particularly with convolutional layers. Therein it appears that the deeper the network layer, the more complex the representations, e.g. from contours to edges, from edges to object parts, and from object parts to objects. This simplified imagination of deep learning refers to the understanding and visualisation of CNNs [35].

2.1.2 Operation Principle

Functionality of a Single Neuron Neural networks are information processing systems. Their structure and functionality are adapted from the human brain and they consist of many parallel working neuron connections. These neurons communicate in the form of activation signals, which are weighted according to their learned importance [36, p. 11].

The simplest structure of a single neuron is a perceptron, see fig. 2.1. The perceptron has $\{x_1, ..., x_N\}$ inputs and one output y_1. Each input is weighted by its corresponding weights $\{w_1, ..., w_N\}$. The sum of all weighted inputs including bias b is forwarded to the activation function ϕ. The activation function produces the non-linearity of each

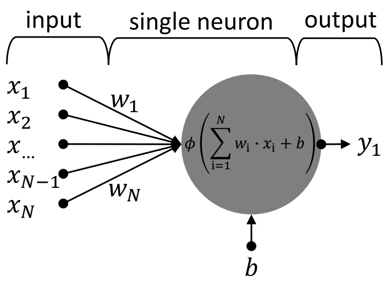


Fig. 2.1 Functionality of a single neuron, a so-called perceptron, within a neural network. The perceptron has N inputs $\{x_1, ..., x_N\}$ and each input has a weight w. The perceptron generates the output y_1 by using a non-linear activation function ϕ for the sum of the weighted input plus a bias b.

neuron by the selection of a non-linear activation function, e.g. a hyperbolic tangent, a sigmoid, or a rectified linear unit function.

Forward Propagation Forward propagation produces the output of a neural network. The input data is gradually processed from network layer to network layer by use of the activation functions and weights. Each network layer thereby reduces the size of its input with simultaneous generalisation.

The forward propagation for a single neuron, cf. fig. 2.1, is defined by

$$y^{\mathrm{l}} = \phi\left(\sum_{\mathrm{i}=1}^{N} w_{\mathrm{i}}^{\mathrm{l}} \cdot x_{\mathrm{i}}^{\mathrm{l}} + b^{\mathrm{l}}\right) \tag{2.1}$$

with N number of contributing neurons $x_{\mathrm{i}}^{\mathrm{l}}$, weighted by $w_{\mathrm{i}}^{\mathrm{l}}$ with an additional bias b^{l}, and separated by independent neural network layers l, cf. [37]. The forward propagation for specific neural network layers, e.g. in a convolutional layer or a pooling layer, is explained in the following subsection 2.1.3.

Backward Propagation Backward propagation is used to update or train the weights of neural network layers. Processing is executed subsequent to the forward propagation after each iteration n and represents the most time-consuming computation step. The backward propagation procedure is described for the above-mentioned single neuron as follows.

First, the error loss

$$\Psi = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}(\hat{y}_{\mathrm{i}} - y_{\mathrm{i}}^{\mathrm{l}})^2 \tag{2.2}$$

is calculated after each iteration. Here, the error loss is given by the Mean Squared Error (MSE) loss function with the target vector $\hat{y}$ and output vector y both with the same length N. Second, the gradients $\Delta w_{\mathrm{i}}^{\mathrm{l}}$ are calculated by use of the chain-rule:

$$\frac{\partial\Psi}{\partial w_{\mathrm{i}}^{\mathrm{l}}} = \frac{\partial y^{\mathrm{l}}}{\partial w_{\mathrm{i}}^{\mathrm{l}}}\frac{\partial \xi^{\mathrm{l}}}{\partial y^{\mathrm{l}}}\frac{\partial\Psi}{\partial \xi^{\mathrm{l}}} \text{ with } \xi^{\mathrm{l}} = \sum_{i=1}^{N} w_{\mathrm{i}}^{\mathrm{l}} \cdot x_{\mathrm{i}}^{\mathrm{l}} + b^{\mathrm{l}}. \tag{2.3}$$

It results by means of simplification, see [34, p. 200-214] for more details,

$$\Delta w_{\mathrm{i}}^{\mathrm{l}} = -\frac{\partial\Psi}{\partial w_{\mathrm{i}}^{\mathrm{l}}} \tag{2.4}$$

as a gradient. The term backward propagation actually describes the determination of the gradients. The weights are updated using another algorithm. In this case, it is the Stochastic Gradient Descent Momentum (SGDM) algorithm, cf. [34, p. 200]. Third, the weights are updated via

$$w_{\mathrm{i}}^{\mathrm{l}}(\mathrm{n}+1) = w_{\mathrm{i}}^{\mathrm{l}}(\mathrm{n}) + \eta \cdot \Delta w_{\mathrm{i}}^{\mathrm{l}}(\mathrm{n}) \tag{2.5}$$

for the next iteration $(\mathrm{n}+1)$ by use of the gradient and the learn rate η, which is explained in section 2.1.4.

2.1.3 Explanation of Neural Network Layers

A neural network, particularly a CNN consists of at least three different layer types: convolutional layers, pooling layers, and fully connected layers. Additionally, a CNN can be extended with more layer types to improve the test results, decrease the training run-time, or enables new network topologies to resolve more complex tasks. In the following, neural network layers are explained briefly that are utilised in this work.

Convolutional Layer The aim of convolutional layers is to learn feature representations[16] from the input image. A convolutional layer is composed of several convolutional filter kernels. Each kernel computes a new convolutional feature map. Particularly, each neuron of a convolutional feature map is locally connected to a region of neighbouring neurons in the previous convolutional layer, depending on the kernel size κ, cf. fig. 2.2. Here, a symmetric kernel size is used. All kernel weights κ^2 of a kernel are shared for all spatial locations (u, v) of the input tensor to produce a new convolutional feature map denoted by k. Moreover, many convolutional layers use a zero pooling extension at the input borders in order to make the feature map size equal to the input tensor size, see fig. 2.2.

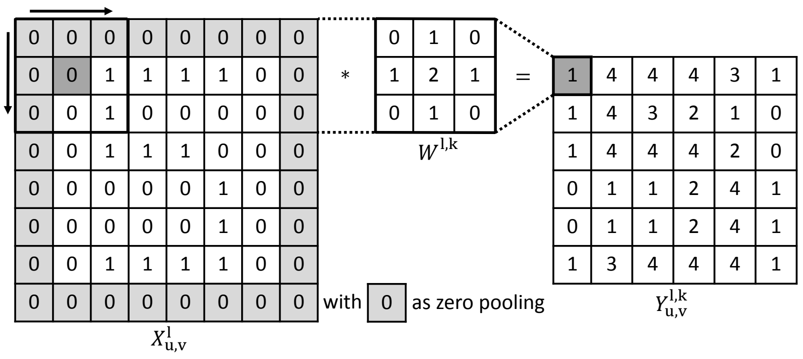


Fig. 2.2 Example of a single feature map convolution. The input tensor feature map $X_{\mathrm{u,v}}^{\mathrm{l}}$ is convolved by a 3 by 3 shared weight filter kernel $W^{\mathrm{l,k}}$ with zero pooling at the borders in order to create the a new convolutional feature map $Y_{\mathrm{u,v}}^{\mathrm{l,k}}$.

[16]A feature representation is understood as an abstract object part, e.g. lines, circles, or edges, cf. also subsection 2.1.1.

A new convolutional feature map

$$Y^{\mathrm{l,k}}_{\mathrm{u,v}} = \sum_{\mathrm{i}=1}^{K_{\mathrm{l}-1}} X^{\mathrm{l,i}}_{\mathrm{u,v}} * W^{\mathrm{l,i,k}} \tag{2.6}$$

is obtained by adding all input tensor feature maps in $X^{\mathrm{l,i}}_{\mathrm{u,v}}$ with $\mathrm{i} \in \{1, ..., K_{\mathrm{l}-1}\}$ at location (u, v), which are convolved with corresponding kernel $W^{\mathrm{l,i,k}}$ of the l-th layer [38]. The number of input tensor feature maps is defined by the number of kernels $K_{\mathrm{l}-1}$ of the previous $(\mathrm{l}-1)$ convolutional layer. Then, a non-linear activation function ϕ, e.g. a rectified linear unit function, is applied to each value of the single convolutional feature map $Y^{\mathrm{l,k}}_{\mathrm{u,v}}$ together with the bias $b^{\mathrm{l,k}}$ of the kernel used. This results in convolutional layer output

$$Z^{\mathrm{l,k}}_{\mathrm{u,v}} = \phi(Y^{\mathrm{l,k}}_{\mathrm{u,v}} + b^{\mathrm{l,k}}). \tag{2.7}$$

This describes the forward propagation in a convolutional layer. The concrete backward propagation for this layer type is described in [34, p. 350-352] and follows the principle in subsection 2.1.2.

Pooling Layer A pooling layer down-samples the output of the previous convolutional layer. It takes an area of features from the previous convolutional layer and selects or generates a new feature, which best represents all taken features. Pooling factor λ^{l} depends on the pooling filter size and the stride length for each resulting pooling location (p, q). The stride length is set to no overlap by default. This means each pooling area is independent. This setting yields by use of a symmetric two-dimensional pooling filter that the pooling factor is equal to the filter size.

Two widely used methods, maximum and average pooling, have high computational efficiency. They are therefore frequently used in pooling layers because they achieve good classification accuracy in open-source databases [39]. The maximum pooling method is defined by the explained default pooling settings as

$$Z^{\mathrm{l,k}}_{\mathrm{p,q}} = \max(\{Z^{\mathrm{l,k}}_{\mathrm{u,v}} : \mathrm{u} = \mathrm{p}, ..., \mathrm{p} + \lambda^{\mathrm{l}} - 1; \mathrm{v} = \mathrm{q}, ..., \mathrm{q} + \lambda^{\mathrm{l}} - 1\}) \tag{2.8}$$

and the average pooling method as

$$Z^{\mathrm{l,k}}_{\mathrm{p,q}} = \frac{1}{(\lambda^{\mathrm{l}})^2} \sum_{\mathrm{u}=\mathrm{p}}^{\mathrm{p}+\lambda^{\mathrm{l}}-1} \sum_{\mathrm{v}=\mathrm{q}}^{\mathrm{q}+\lambda^{\mathrm{l}}-1} Z^{\mathrm{l,k}}_{\mathrm{u,v}}. \tag{2.9}$$

Fully Connected Layer A fully connected layer is composed by a parallel arrangement of single neurons or perceptrons. A conjunction of one or more fully connected layers is called a Multi-Layer Perceptron (MLP) in the reference literature [40]. In a fully

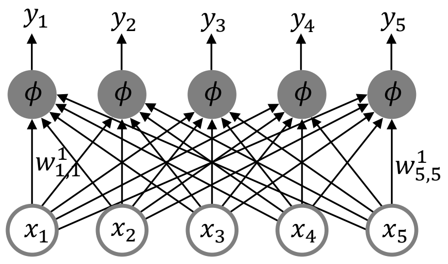


Fig. 2.3 Extract of a Multi-Layer Perceptron (MLP). It contains one fully connected layer with five inputs x_i and five outputs y_j. ϕ are the non-linear activation functions and $w^1_{\mathrm{i,j}}$ are the connection weights.

connected layer, all inputs are connected with their weights to every single neuron, cf. fig. 2.3, which shows a fully connected layer with five inputs $\{x^\mathrm{l}_\mathrm{i}, ..., x^\mathrm{l}_{N_\mathrm{i}}\}$ and outputs $\{y^\mathrm{l}_\mathrm{j}, ..., y^\mathrm{l}_{N_\mathrm{j}}\}$ and their weights $w^\mathrm{l}_{\mathrm{i,j}}$. The fully connected layer input x^l_i is mostly the output of the last pooling layer $Z^{\mathrm{l,k}}_{\mathrm{p,q}}$ for the first fully connected layer in a CNN. For this all output locations (p, q) of each feature map k in the pooling layer have to be flattened from a tensor to a vector. It applies $\mathrm{i} \in \{1, ..., P \cdot Q \cdot K\}$ with K feature maps and P, Q as the tensor height and width.

Dropout Layer Dropout layers are used during the training process and have two benefits. First, they improve neural networks by reducing over-fitting[17]. Second, they improve the performance of neural networks independently from the dataset used. This is achieved by breaking up redundant adaptations between fully connected layers, which are inconvenient for generalising unseen data [41].

A typical configuration places the dropout layer in front of a fully connected layer. The dropout layer itself is modelled by a Bernoulli distribution $\mathrm{B}(p_B)$. Each input signal of a neuron is multiplied by an independent random value $\chi^\mathrm{l}_\mathrm{i}$, which has the probability p_B of being one. It applies for the output

$$y^\mathrm{l}_\mathrm{j} = \phi\left(\sum_{\mathrm{i}=1}^{N_\mathrm{i}} w^\mathrm{l}_{\mathrm{i,j}} \cdot x^\mathrm{l}_\mathrm{i} \cdot \chi^\mathrm{l}_\mathrm{i}\right) \tag{2.10}$$

in a fully connected layer with an upstream dropout layer with

$$\chi^\mathrm{l}_\mathrm{i} \sim \mathrm{B}(p_B). \tag{2.11}$$

Batch Normalisation Layer A batch normalisation layer is deployed before the activation function of a convolutional or fully connected layer. The layer normalises the batch

[17]Over-fitting means that the CNN has perfectly learned the training dataset, but is not able to successfully transfer the learned features on an unseen test dataset.

input of the dataset. Thus, the activation functions always react to the same input distribution because otherwise the distribution of each layer's inputs would changes during the CNN training process. This is also known as the internal co-variance shift problem, which is addressed through a batch normalisation layer. In consequence, a batch normalisation layer decreases the training run-time and allows a greater degree of freedom for the initial learning rate parameter [42].

The batch normalisation algorithm follows two steps: (1) the normalisation and (2) the scale and shift. First, all activation inputs $Y_{\mathrm{p,q}}^{\mathrm{l,k,(b)}}$ of a batch size B with $\mathrm{b} \in \{1, ..., B\}$ are normalised to

$$\hat{Y}_{\mathrm{p,q}}^{\mathrm{l,k,(b)}} = \frac{Y_{\mathrm{p,q}}^{\mathrm{l,k,(b)}} - \mu_{(B)}}{\sqrt{\sigma_{(B)}^2 + \epsilon}}. \tag{2.12}$$

in terms of the batch mean $\mu_{(B)}$ and the batch variance $\sigma_{(B)}^2$ within a layer l. The technical parameter ϵ is set to $0 < \epsilon \ll 1$ to avoid dividing through zero. The batch mean $\mu_{(B)}$ and the batch variance $\sigma_{(B)}^2$ are calculated with $\mathrm{l} = 1$ as follows:

$$\mu_{(B)} = \frac{1}{BKPQ} \sum_{\mathrm{b}=1}^{B} \sum_{\mathrm{k}=1}^{K} \sum_{\mathrm{p}=1}^{P} \sum_{\mathrm{q}=1}^{Q} Y_{\mathrm{p,q}}^{\mathrm{k,(b)}}, \tag{2.13}$$

$$\sigma_{(B)}^2 = \frac{1}{BKPQ} \sum_{\mathrm{b}=1}^{B} \sum_{\mathrm{k}=1}^{K} \sum_{\mathrm{p}=1}^{P} \sum_{\mathrm{q}=1}^{Q} (Y_{\mathrm{p,q}}^{\mathrm{k,(b)}} - \mu_{(B)})^2. \tag{2.14}$$

Second, the normalised activation inputs are scaled and shifted with the parameters θ and β, which are to be learned:

$$\tilde{Y}_{\mathrm{p,q}}^{\mathrm{k,(b)}} = \hat{Y}_{\mathrm{p,q}}^{\mathrm{k,(b)}} \cdot \theta + \beta. \tag{2.15}$$

The parameter θ and β are updated via

$$\frac{\partial \Psi}{\partial \theta} = \sum_{\mathrm{b}=1}^{B} \sum_{\mathrm{k}=1}^{K} \sum_{\mathrm{p}=1}^{P} \sum_{\mathrm{q}=1}^{Q} \frac{\partial \Psi}{\partial \tilde{Y}_{\mathrm{p,q}}^{\mathrm{k,(b)}}} \cdot \hat{Y}_{\mathrm{p,q}}^{\mathrm{k,(b)}} \text{ and} \tag{2.16}$$

$$\frac{\partial \Psi}{\partial \beta} = \sum_{\mathrm{b}=1}^{B} \sum_{\mathrm{k}=1}^{K} \sum_{\mathrm{p}=1}^{P} \sum_{\mathrm{q}=1}^{Q} \frac{\partial \Psi}{\partial \tilde{Y}_{\mathrm{p,q}}^{\mathrm{k,(b)}}} \tag{2.17}$$

through backward propagation of the error loss Ψ, explained in sec. 2.1.2, during the training process, cf. [42]. The gradients are used to improve iteratively θ and β as explained in equation 2.5. When the training process is complete, the parameters $\mu_{(B)}$ and $\sigma_{(B)}^2$ will be generated and stored for the entire training dataset. These parameters are used to normalise new unseen test datasets identical to the training dataset.

Merge Layer The purpose of merge layers is to construct more complex network topologies. They enable to combine or fuse two or more input data streams. There are several kinds of merge layers. In principle, all merge layers are formulated via mathematical operators which combine multiple data streams. The most famous one is the concatenation layer in equation 2.18, which simply chains features in a specified dimension:

$$X^{\mathrm{k}} = X^{\mathrm{k}_1} \frown X^{\mathrm{k}_2} \text{ with } \mathrm{k} \in \{\mathrm{k}_1, \mathrm{k}_2\} \tag{2.18}$$

Moreover, element-wise multiplication, addition, and subtraction layers exist, cf. [43].

Disadvantages of merge layers are that more effort is needed for network specification, e.g. for the hyper parameter setup (see next subsection 2.1.4) and that the complexity of backward propagation increases the computational outlay of training run-time.

Long Short Term Memory Layer A Long Short Term Memory (LSTM) layer processes data time series or frame sequences. The LSTM layer is designed as a constant-error-carousel that is self-connected to its output. The resulting layer architecture is also called a *memory cell*, cf. [44].

Figure 2.4 illustrates the architecture of an LSTM layer in the form of a block diagram. This LSTM layer is designed to process the same input and output data format as for a fully connected layer. The input and output vectors are $x(\mathrm{f})$ and $y(\mathrm{f})$. Here, the time

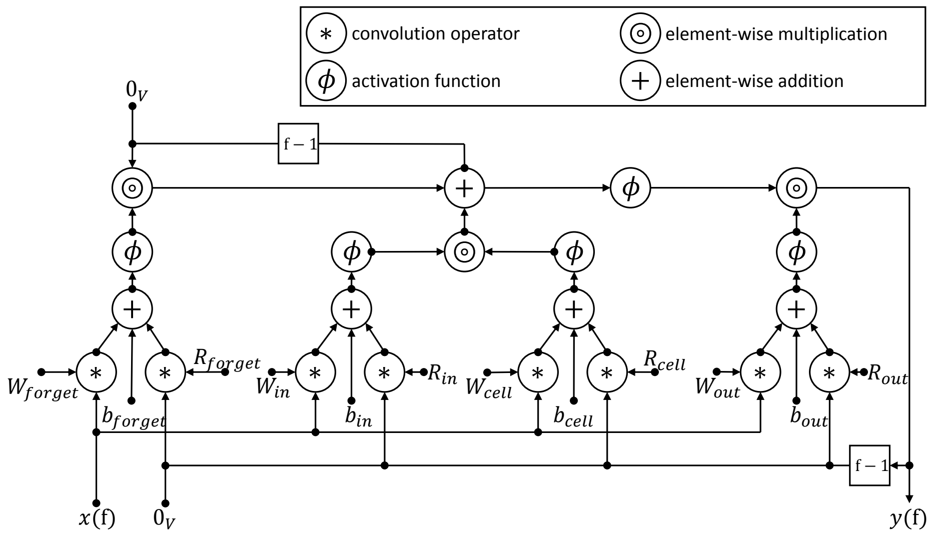


Fig. 2.4 Block diagram of a Long Short Term Memory (LSTM) layer. It uses the input $x(\mathrm{f})$ and previous output $y(\text{f-1})$ for each frame iteration f to generate the output $y(\mathrm{f})$. It consists of four gates: forget gate (*forget*), input gate (*in*), cell gate (*cell*), and output gate (*out*). It is reinitialised with vectors 0_V filled with zeros for each new iteration cycle.

steps are represented by frame index f corresponding to the current input image of a CNN. The LSTM layer consists of a forget (*forget*), input (*in*), cell (*cell*), and output (*out*) gate. All these gates have as inputs the current input vector $x(\mathrm{f})$ and the previous output vector $y(\mathrm{f}-1)$. Each gate is trained by the convolution and addition of the inputs with the weight matrices W, recurrent weight matrices R, and bias b (equivalent to the forward propagation in section 2.1.2). Each gate output is also activated by a non-linear activation function ϕ.

The cell gate is the core of the LSTM layer. It is regulated by the input gate by means of the element-wise multiplication of their outputs. The forget gate controls which values can pass by addition. Technically, it is self-connected via element-wise multiplication of its previous decision at frame $\mathrm{f}-1$. The LSTM output $y(\mathrm{f})$ is activated again and passes the output gate by means of another regulation by an element-wise multiplication, see figure 2.4.

In a new iteration cycle, the LSTM layer is reinitialised by use of vectors 0_V filled with zeros and the LSTM process described starts again.

2.1.4 Hyper Parameter Setup

Hyper parameters are parameters that have to be defined manually by the researcher knowledge before the training process starts, cf. [45]. Hyper parameter setup is mostly very time consuming and the hyper parameters have a direct influence on three deep learning properties: hardware resources, training run-time, and test results. These deep learning properties also influence each other, see figure 2.5. In most cases, the hardware resources (computing power and memory storage) are limited. The training run-time depends on the available hardware resources, such as the number of parallel GPUs, their graphical processing power, graphical memory storage, and of course the hyper parameters. The deep learning property test results especially depend on the hyper parameters and not exactly on the training run-time. This means that a CNN

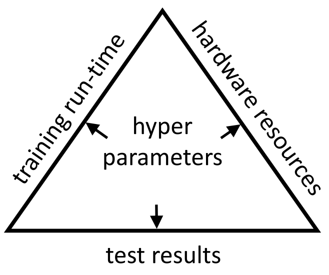


Fig. 2.5 Relation of CNN hyper parameters and three deep learning properties. The hyper parameter setup influences the training run-time, the test results, and the hardware resources of a CNN.

with badly conditioned hyper parameters could evoke a very long training run-time but achieve poor test results.

Important hyper parameters are the convolutional kernel size κ (symmetric two-dimensional filter kernel), the number of convolutional feature maps K, and the pooling factor λ^{l} of the current layer l to set up a CNN topology. These three hyper parameters determine the number of neurons in the next convolutional layer and the number of weights in the current convolutional layer, cf. [46]. The number of neurons is given by

$$N_N = \frac{P \cdot Q}{(\lambda^{\mathrm{l}})^2} \cdot K \tag{2.19}$$

with P and Q as the height and width of the input tensor for the convolutional layer. The number of weights is defined by

$$N_W = \kappa^2 \cdot K. \tag{2.20}$$

The number of neurons, the number of weights, and the number of convolutional layers impact on run-time (the larger the CNN is, the longer the training run-time takes) as well as the test results. Note that on the one hand, if the number of weights is too small in the CNN, the CNN will not learn any generalisation. On the other hand, if the number of weights is too big in the CNN, the CNN will be over-fitted after the training process such that the transferability of the results is lost.

Furthermore, three important hyper parameters are introduced below to setup the training process of a CNN:

Initial Learn Rate The initial learn rate η is the most challenging parameter. In a common MLP, the initial learn rate is by default one, but by using CNNs, the range is often between $[10^{-5}, 10^{-2}]$. The best practice is to set a minimum and a maximum initial learn rate value, monitor the CNN error loss of each training process, and improve the value by using the bisection method after each training.

Moreover, learn rate decay is a technique to improve the training process by means of step-wise reduction of the learn rate, e.g. by a factor of two, after a defined number of epochs. This technique is also known as the L2 regulation. Note that on the one hand, if the learn rate is very small, the loss of the CNN will decrease only after a long run-rime or possibly never. On the other hand, if the learn rate is too large, the CNN error loss will diverge and the training process should be stopped immediately.

Batch Size The batch size[18] hyper parameter is a technical parameter and is limited by the availability of memory storage on GPUs. This guarantees a maximum utilisation

[18]Bach size is also mentioned as mini batch size in literature.

of the hardware resources. Unless if the mini batch size is too small (only a handful of input images), the CNN will not learn any generalisation and the CNN error loss will not start to decrease.

Number of Epochs This hyper parameter specifies how often the training dataset should be processed via the CNN. The best practice is to set up a large number of epochs using early stopping by means of a validation dataset. Otherwise, if the training process processes the training dataset with too many epochs, the CNN will be over-fitted after the training process.

Furthermore, the number of epochs N_E, the batch size B, and the number of training dataset images N_D influence the number of iterations $N_I = N_E \cdot \frac{N_D}{B}$. After each iteration, the kernel weights of a CNN are updated.

The goal should be to achieve the best possible result with a reasonable training run-time and the given hardware resources. To achieve this, it should be started from a minimal hyper parameter setup for a CNN topology at the beginning. Moreover, there are approaches for automatically finding the optimal hyper parameter setup by using efficient search algorithms [47], [48] or advising the initial hyper parameter setup based on a previous brute force optimisation study [49]. However, the automatic setting of CNN hyper parameters is still an open-research question.

2.1.5 Normalisation in Convolutional Neural Networks

Normalisation has become more important during the usage of CNNs in the last few years, since normalisation has enabled a faster training process and a greater generalisation of a CNN while a less careful hyper parameter initialisation.

The four normalisation types are explained below. They all have data normalisation in common to make updating the weights more efficient for each iteration. Moreover, a dropout layer as explained in subsection 2.1.3, which is actually a regularisation layer, also realises a type normalisation. Instead of data normalisation, it directly normalises the weights of the following fully connected layer.

Zero Normalisation This normalisation type normalises the entire dataset before the training. The global mean of all input images is determined and is subtracted from each image value separately for each image channel. This is the easiest normalisation type and is applied normally to every CNN.

Batch Normalisation Batch normalisation normalises the single mini batches with each other during training. Batch normalisation needs its own network layer for implementation and was explained in detail in subsection 2.1.3.

Layer Normalisation Layer normalisation works in principle as well as batch normalisation on layer level. It has the adjustment that all feature maps of an input image in a selected convolutional layer are normalised with each other, cf. [50]. This is applied to each input image.

Instance Normalisation Instance normalisation is a sub-type of layer normalisation, cf. [50]. It normalises in the same way as layer normalisation, but with the distinction that only certain and not all convolutional feature maps are normalised with each other.

2.1.6 Definition of Deep Fusion

Deep Fusion belongs to the research field of artificial intelligence and is a sub-aspect of deep learning. The definition of the term deep fusion is not consistent in the reference literature. However, all scientific publications have in common that deep fusion is about deep information fusion and it describes a technique to integrate several data streams together in a CNN, cf. [51], [52]. Deep fusion can take place in an early or late processing step in the CNN topology, which is usually clarified by additional experiments, depending on the use case. Each deep fusion approach can be broken down into the following categories:

Fusion Technique There are two main fusion techniques: (1) multi-feature fusion and (2) heterogeneous data fusion. The difference between both is conditioned by the input data. Multi-feature fusion uses only one data source, e.g. image data, and splits them within the CNN in different data streams, which are then fused together again in a later processing step, cf. [53]–[55]. By contrast, heterogeneous data fusion takes several data streams, e.g. image data and metadata, as input data and fuses them in the CNN, see also subsection 4.2.3.

Fusion Method The fusion method also conditions the fusion operator used. Two methods exist [56]: (1) direct fusion and (2) aggregation fusion. Direct fusion presumes that the different data streams are adapted to the same size. Here, fusion operators like an element-wise addition, element-wise multiplication, or a convolution are used. Aggregation fusion do not need data streams of the same size. Hence, fusion operators like maximisation, minimisation, or concatenation, which aggregate the data streams, are used, cf. [57].

Fusion Level In general, deep fusion is divided into three fusion levels: (1) raw level, (2) feature level, and (3) object level. The first level describes the fusion of raw data streams, e.g. the deep fusion of camera images and disparity maps to create high resolution depth maps, e.g. in [58] and [59]. The second level describes the fusion of previously generated features. This fusion level is often used in combination with the multi-feature fusion technique. The third level describes the fusion of already recognised objects. For example, this fusion level is employed in order to verify detected objects by a camera and RaDAR sensor, cf. [60].

Note that the deep metadata fusion approach as proposed in chapter 4 can be categorised as a direct deep feature fusion with heterogeneous data in this work.

2.2 Method of Inverse Perspective Mapping

Inverse Perspective Mapping (IPM) describes in general an image transformation. It is part of the linear algebra in the field of perspective projection. An image is transformed from its origin plane into another plane inside a three-dimensional coordinate system, while the point of view stays the same. Other known terms are bird's-eye-view or top view, which describe an often used perspective projection: The point of view of the camera is virtually transformed vertically above the ground.

For this work, I developed an IPM method to create top view images from an ego-vehicle camera. The aim is to generate an IPM full panorama image of the complete intersection scenario by stitching single top view images of the same sequence together. Unfortunately, only the ego-vehicle camera yaw angle is available, but the pitch and roll angles are missing. Thus, the contribution of this approach is a procedure which compensates nick angular movements between stitched IPM frames by means of the determination of the image horizon in its three-dimensional point cloud image. The roll angle is neglected in the approach. Moreover, the main extensions of this IPM approach in contrast to a conventional IPM transformation, cf. [61], [62], are:

- A flat world assumption function to convert quickly between the camera coordinate system and the world coordinate system.
- A mean absolute error optimisation to find the best match between the previous and next stitched top view images.
- A cross-correlation back-correcting to minimise the deviation between the generated IPM full panorama image and each single top view image.

This IPM approach is only real-time capable with high computational outlay because of the pitch angle compensation and the mentioned extensions. However, the IPM method is also used for efficient and offline IPM lane line marking annotations, see upcoming subsection 2.2.5.

2.2.1 Image Horizon Estimation

A three-dimensional point cloud image is used to estimate the image horizon u_H, which is used to compensate the unknown pitch angle of the ego-vehicle. The image horizon (marked as red line in fig. 2.6) is calculated for each camera image and constitutes the key parameter for each IPM transformation into a top view image. It is formed from the intersection line of the two planes E_{UV} and E_{road} and represents the constant u-intercept of the intersection line in the image coordinate system. E_{UV} is the UV-plane equation of the camera coordinate system. The plane E_{road} is the recognised road part in the camera image that is projected into the top view image. It is illustrated as a blue

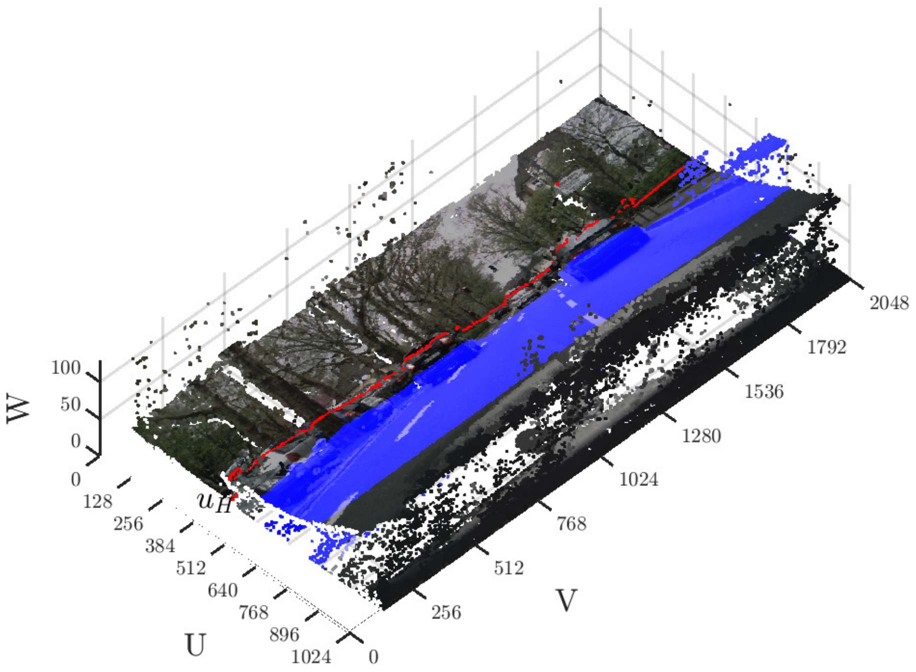


Fig. 2.6 Illustration of a three-dimensional point cloud image $D_{\mathrm{u,v,w}}$. The point cloud image has in its dimensions U and V the camera image $I_{\mathrm{u,v}}$ and in the dimension W the disparity map of the camera image. The road plane E_{road} (blue surface) is used to determine the virtual horizon u_H (red horizontal line).

transparent surface in fig. 2.6 and is determined by use of the RANSAC algorithm[19]. The RANSAC algorithm randomly iterates through many possible planes of E_{road} to find the most robust match between the estimated plane and the point cloud image pixels [63]. The RANSAC algorithm needs as input the three-dimensional point cloud image stored in $D_{\mathrm{u,v,w}}$ and the initial search plane E_{WU}, which represents the WU-plane equation of the camera coordinate system. The parameters of the RANSAC algorithm are: number of iterations (500), maximum error distance to decide between inlier and outlier pixels (2 px), and the percentage of correctly fitted pixels to abort early the search (90 %).

Figure 2.6 shows an example of a three-dimensional point cloud image $D_{\mathrm{u,v,w}}$. This image is created by use of the (left) camera image $I_{\mathrm{u,v}}$ (with the row position u and column position v) and the disparity map of the left and right camera image with a base width of 0.2 m. The disparity map is generated by means of the Semi Global Matching (SGM) stereo processing algorithm, cf. [64], [65]. It represents the image depth in pixels, which is plotted in dimension W of fig. 2.6.

2.2.2 Flat World Assumption

The IPM transformation requires a function that represents a flat world assumption for the road area in front of the ego-vehicle. This enables a fast and consistent conversion between the image row index u and the longitudinal world distance z(u). This function can be derived from figure 2.7. The longitudinal world distance

$$\mathrm{z(u)} = \tan(\alpha') \cdot h_{cam} \tag{2.21}$$

is calculated by using the angle α' and the world camera height $h_{cam} = 1.24$ m. The angle α' is specified by the alternate angle α:

$$\alpha' = \frac{\pi}{2} - \alpha. \tag{2.22}$$

The angle

$$\alpha = \tan^{-1}\left(\frac{(u_H - \mathrm{u}) \cdot \rho}{f}\right) \tag{2.23}$$

is defined by the difference between the image horizon u_H, the image row index u, and the two fixed camera parameters: pixel density $\rho = 2\ \frac{\mu\mathrm{m}}{\mathrm{px}}$ and focal length $f = 4.8$ mm. The final equation 2.24 results from inserting the equations 2.21, 2.22, and 2.23 into each other:

$$\mathrm{z(u)} = \tan\left(\frac{\pi}{2} - \tan^{-1}\left(\frac{(u_H - \mathrm{u}) \cdot \rho}{f}\right)\right) \cdot h_{cam}. \tag{2.24}$$

[19]RANSAC stands for Random Sample Consensus.

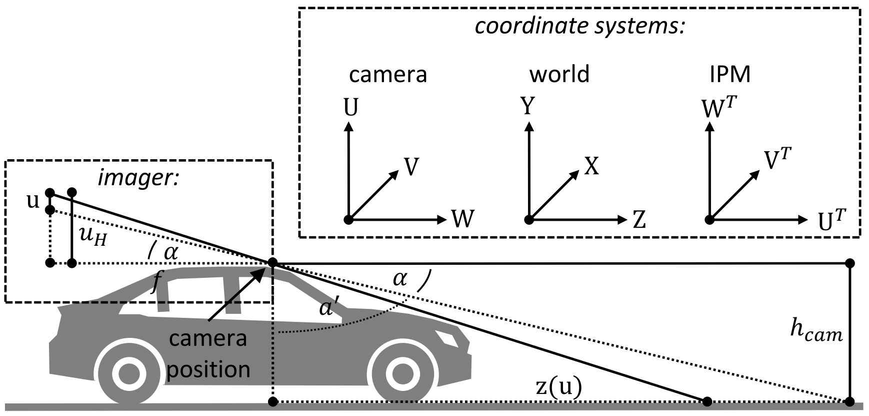


Fig. 2.7 Explanation of the flat world assumption function and its corresponding coordinate systems. The projection into other coordinate systems uses the camera view only.

The flat world assumption function, equation 2.24, describes a simplification of the road plane E_{road} depending on the U-dimension in the three-dimensional point cloud image, since it relies on the image horizon u_H.

The lateral world distance

$$\mathrm{x(v)} = \frac{\mathrm{v} \cdot \rho}{f} \cdot \mathrm{z(u)} \qquad (2.25)$$

is calculated by means of the standard pin-hole camera model[20]. This equation uses the image column index and the previously determined longitudinal world distance represented by v and z(u).

Moreover, figure 2.7 displays three coordinate systems: the camera coordinate system (U, V, W), the world coordinate system (Y, X, Z), and the IPM coordinate system (W^T, V^T, U^T). All coordinate systems use Cartesian coordinates. These coordinate systems enable the IPM transformation from a camera image $I_{\mathrm{u,v}}$ into a top view image $J_{\mathrm{u}^T,\mathrm{v}^T}$ by means of the flat world assumption made and the image horizon estimation. The indices u^T and v^T in $J_{\mathrm{u}^T,\mathrm{v}^T}$ are the row and column indices of the transformed top view image. The IPM method is split in two main algorithms, the IPM full panorama image stitching and IPM top view image back-correction, which are introduced in subsections 2.2.3 and 2.2.4 below.

2.2.3 Full Panorama Image Stitching

The IPM full panorama image stitching procedure is described in algorithm 1 for one intersection scenario or sequence. The algorithm 1 is now explained line by line:

The procedure loops each camera image $I^{\mathrm{f}}_{\mathrm{u,v}}$ by the frame index f of all frames F to generate the IPM full panorama image $O_{\mathrm{u}^T,\mathrm{v}^T}$, see line 3. In the loop, the

[20]The standard pin-hole camera model based on the theorem of intersecting lines.

Algorithm 1 IPM full panorama image stitching procedure.

```
1: input: $I^{\mathrm{f}}_{\mathrm{u,v}}$: camera image with frame f,
          $D^{\mathrm{f}}_{\mathrm{u,v,w}}$: point cloud tensor of disparity map,
          $ROI^{\mathrm{f}}_{\mathrm{u,v}}$: region of interest from $I^{\mathrm{f}}_{\mathrm{u,v}}$,
          $E_{\mathrm{UV}}$: UV-plane in image coordinate system,
          $E_{\mathrm{WU}}$: WU-plane in image coordinate system,
          $M^{\mathrm{f}}_{rot}$: rotation matrix,
          $M^{\mathrm{f}}_{transl}$: translation matrix,
          $IPM_{\mathrm{config}}$: fixed IPM configuration parameters.
2: output: $O_{\mathrm{u}^T,\mathrm{v}^T}$: IPM full panorama image.
3: for f = 1:$F$ do
4:   $E_{road} \leftarrow f_{\mathrm{RANSAC}}(D^{\mathrm{f}}_{\mathrm{u,v,w}},\ E_{\mathrm{WU}})$    // search best road plane
5:   $u^{\mathrm{f}}_H \leftarrow E_{road} \cap E_{\mathrm{UV}}$    // calculate intersection line
6:   if f > 1 then
7:     for i = -5:5 do
8:       $H \leftarrow f_{\mathrm{TRANSFORM}}(u^{\mathrm{f}}_H + \mathrm{i},\ IPM_{config})$    // create transformation matrix
9:       $J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow I^{\mathrm{f}}_{\mathrm{u,v}} \cdot H\ \forall(\mathrm{u,v}) \in ROI^{\mathrm{f}}_{\mathrm{u,v}}$    // transform camera image
10:      $J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T} \cdot M^{\mathrm{f}}_{transl} \forall(\mathrm{u}^T,\mathrm{v}^T)$    // translate top view image
11:      $J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T} \cdot M^{\mathrm{f}}_{rot} \forall(\mathrm{u}^T,\mathrm{v}^T)$    // rotate top view image
12:      $\Delta_{\mathrm{i}} \leftarrow f_{\mathrm{MAE}}(J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T},\ J^{\mathrm{f\text{-}1,i_{min}}}_{\mathrm{u}^T,\mathrm{v}^T})$    // compare images by MAE
13:    end for
14:    $\mathrm{i_{min}} \leftarrow \min(\{\Delta\})$    // select best top view image
15:    $O_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow O_{\mathrm{u}^T,\mathrm{v}^T} \oplus J^{\mathrm{f,i_{min}}}_{\mathrm{u}^T,\mathrm{v}^T}$    // stitch top view image
16:  else
17:    $H \leftarrow f_{\mathrm{TRANSFORM}}(u^{\mathrm{f}}_H,\ IPM_{config})$    // create transformation matrix
18:    $J^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow I^{\mathrm{f}}_{\mathrm{u,v}} \cdot H\ \forall(\mathrm{u,v}) \in ROI^{\mathrm{f}}_{\mathrm{u,v}}$    // transform camera image
19:    $O_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow J^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T}$    // initialise IPM panorama image
20:  end if
21: end for
```

RANSAC algorithm is used to estimate the road plane E_{road} of the disparity map point cloud image $D^{\mathrm{f}}_{\mathrm{u,v,w}}$ by means of the initial plane E_{WU} in the WU-plane of the image coordinate system, see line 4. In line 5, the intersection line between the planes E_{road} and E_{UV} is calculated to determine the image horizon u^{f}_H of the camera image $I^{\mathrm{f}}_{\mathrm{u,v}}$, cf. subsection 2.2.1.

Then, a case differentiation is made depending on the frame index f. In case of f = 1, the procedure initialises the IPM full panorama image with the first top view image, see lines 17-19, which are similar to lines 8, 9 and 15. In case of f > 1, the procedure loops variations of the virtual horizon with $u_H^{\mathrm{f}} \in \{-5, ..., 5\}$ pixels to optimise the IPM transformation into the top view image $J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T}$, see lines 7 to 13.

A perspective projection transformation matrix H is generated for each variation, see line 8. The matrix H is calculated by means of the four top view image corner points $(\{\mathrm{u}_\mathrm{j}^T, \mathrm{u}_\mathrm{j}^T\} : \mathrm{j} \in \{1, ..., 4\})$. These points correspond to the four regions of interest corner points $(\{\mathrm{u}_\mathrm{j}, \mathrm{u}_\mathrm{j}\} : \mathrm{j} \in \{1, ..., 4\})$, displayed in figure 2.8. These points are used to solve the equation[21]

$$\begin{bmatrix} \mathrm{u}_\mathrm{j}^T \\ \mathrm{v}_\mathrm{j}^T \\ 0 \end{bmatrix} = H \cdot \begin{bmatrix} \mathrm{u}_\mathrm{j} \\ \mathrm{v}_\mathrm{j} \\ 1 \end{bmatrix} . \tag{2.26}$$

The regions of interest corner points are calculated by rearranging equation 2.24 and 2.25 to u and v and by plugging in the fixed IPM transformation parameters for z(u) and x(v). The fixed IPM transformation parameters IPM_{config} are:

- $\mathrm{z}_1(\mathrm{u}) = 50$ m "world top view longitudinal top"
- $\mathrm{z}_2(\mathrm{u}) = 10$ m "world top view longitudinal bottom"
- $\mathrm{x}_1(\mathrm{v}) = 10$ m "world top view lateral right"
- $\mathrm{x}_2(\mathrm{v}) = -10$ m "world top view lateral left"

Subsequently, the camera image is transformed into the top view image $J^{\mathrm{f,i}}_{\mathrm{u}^T,\mathrm{v}^T}$ by multiplying each image pixel (u,v) inside the defined region of interest image $ROI^{\mathrm{f}}_{\mathrm{u,v}}$ with the transformation matrix H, see line 9. An example region of interest image is shown in figure 2.8, which corresponds to figure 2.6.

The resulting top view image is scaled on the global top view length and width by use of a pixel size of 0.02 m. It is translated and rotated, see line 10-11, by use of the translation matrix

$$M^{\mathrm{f}}_{transl} = \begin{bmatrix} \omega^{\mathrm{f}}_{U^T} & 0 \\ 0 & \omega^{\mathrm{f}}_{V^T} \end{bmatrix} \tag{2.27}$$

and rotation matrix

$$M^{\mathrm{f}}_{rot} = \begin{bmatrix} \cos(\gamma^{\mathrm{f}}) & -\sin(\gamma^{\mathrm{f}}) \\ \sin(\gamma^{\mathrm{f}}) & \cos(\gamma^{\mathrm{f}}) \end{bmatrix} . \tag{2.28}$$

The ego-vehicle yaw rate around the rear axle is given by γ^{f}. Note that the rotation centre lies out of the top view image, because γ is centred on the rear axle of the

[21] The equation 2.26 would normally be constrained for H with nine variables. Two variables are freely select-able for scaling, however, two variables are firmly set to zero, and another variable is always one.

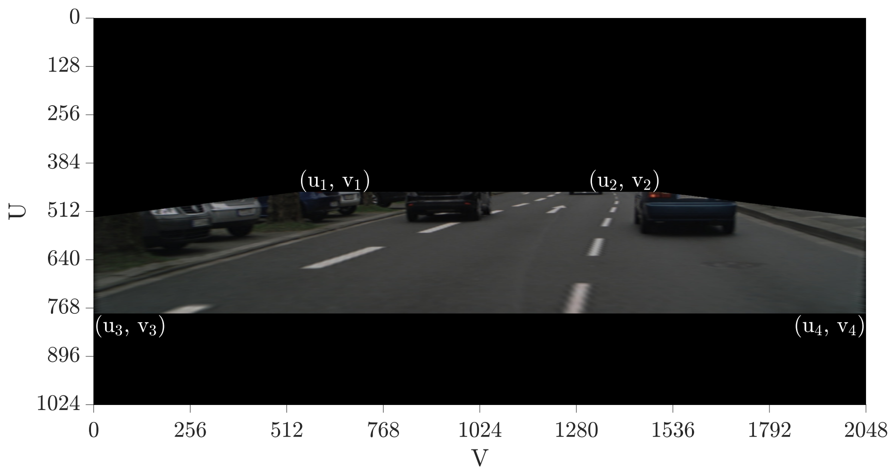


Fig. 2.8 Region of interest image $ROI^{\mathrm{f}}_{\mathrm{u,v}}$ of the camera image $I^{\mathrm{f}}_{\mathrm{u,v}}$ with the four region of interest corner points.

ego-vehicle. The ego-vehicle speed is represented by $\omega^{\mathrm{f}}_{U^T}$ and $\omega^{\mathrm{f}}_{V^T}$ for the U^T and V^T dimension.

The current top view image is compared to the previous one by means of the Mean Absolute Error (MAE), see line 12. Selected is the top view image $J^{\mathrm{f,i_{min}}}_{\mathrm{u}^T,\mathrm{v}^T}$ with the minimal derivation, which is stitched on the IPM full panorama image, see line 14-15.

As a final result, the IPM full panorama image $O_{\mathrm{u}^T,\mathrm{v}^T}$ is gained. Figure 2.9 shows the IPM full panorama image, which belongs to the point cloud image $D_{\mathrm{u,v,w^f}}$ in figure 2.6 as well as to the region of interest image $ROI^{\mathrm{f}}_{\mathrm{u,v}}$ in figure 2.8. The blue surface illustrates the position where this top view image was stitched after transformation, translation, rotation, and optimisation of the camera image.

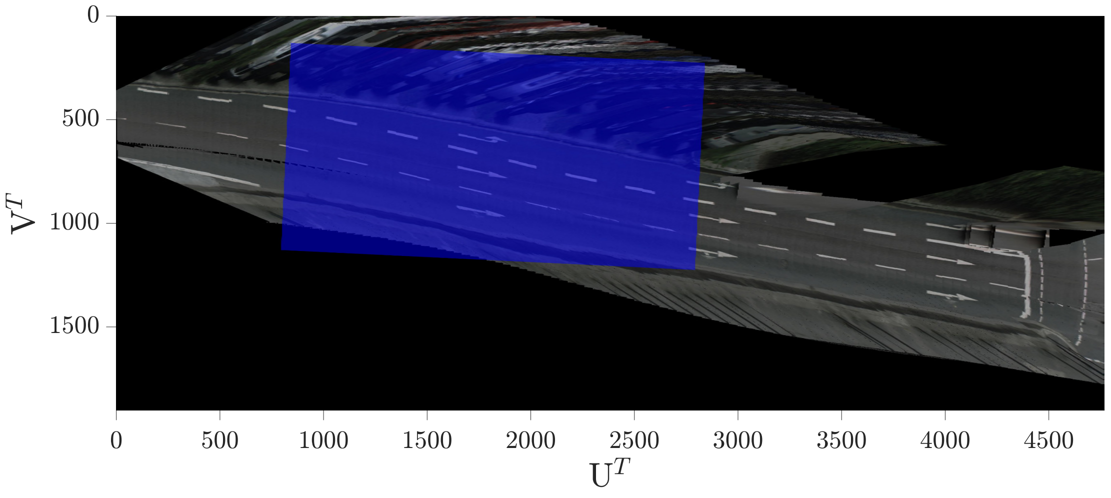


Fig. 2.9 IPM full panorama image with an example position of a stitched top view image. The blue surface corresponds to the blue one in figure 2.6.

2.2.4 Top View Image Back-Correction

The IPM back-correction is an extension of the IPM full panorama image stitching procedure, cf. algorithm 1. The aim is to improve the IPM transformation into a top view image by using the already existing IPM panorama top view image. Hence, this procedure can only be applied after the IPM full panorama image stitching procedure is finished and the IPM full panorama image is created.

The IPM back-correction procedure is displayed in algorithm 2. First, an IPM panorama top view image $O^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T}$ is cut out and rotated back to the original size of a single top view image for a frame f, see line 3. An example of an extracted IPM panorama top view image is shown in fig. 2.10 (b) which conforms to the blue surface in figure 2.9. Second, a variation of the virtual horizon v^{f}_H is looped with $\mathrm{i} \in \{-10, ..., 10\}$ pixels, see line 4-8. The top view images $J^{\mathrm{i}}_{\mathrm{u}^T,\mathrm{v}^T}$ are generated by using the likewise newly created transformation matrices H in the loop, see line 5-6. The camera image transformation into a top view image works identically to the transformation described in line 8-9 of algorithm 1, cf. subsection 2.2.3. Third, the new top view images are compared against the previously extracted IPM panorama top view image by using the Cross-Correlation Function (CCF), see line 7. The CCF iterates the image by a row-wise shift. As a return value, the maximum result of all iterations is selected. The algorithm selects the highest cross-correlation result of all variations, which determines the improved top view image, see line 9-10.

The advantage of this procedure is that the top view images are as similar as possible

Algorithm 2 IPM top view image back-correction procedure.

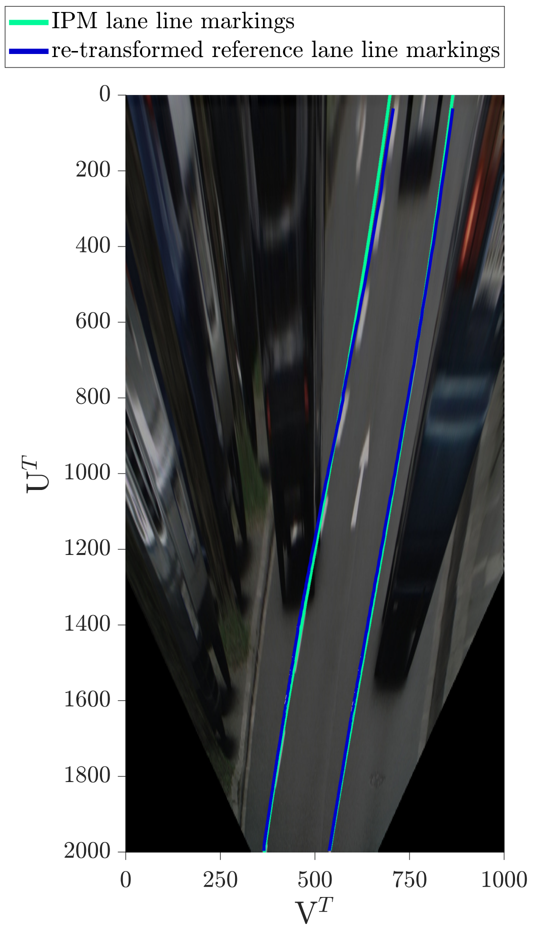
```
 1: input:   $O_{\mathrm{u}^T,\mathrm{v}^T}$: IPM full panorama image,
             $I^{\mathrm{f}}_{\mathrm{u,v}}$: camera image with frame f,
             $IPM_{\mathrm{config}}$: fixed IPM configuration parameters,
             $u^{\mathrm{f}}_H$: initial virtual horizon of frame f
 2: output:  $J^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T}$: improved top view image of frame f.
 3: $O^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T} = f_{EXTRACT}(O_{\mathrm{u}^T,\mathrm{v}^T},\ \mathrm{f})$          // extract IPM panorama top view image
 4: for i = -10:10 do
 5:     $H \leftarrow f_{\mathrm{TRANSFORM}}(u^{\mathrm{f}}_H + \mathrm{i},\ IPM_{\mathrm{config}})$    // create transformation matrix
 6:     $J^{\mathrm{i}}_{\mathrm{u}^T,\mathrm{v}^T} \leftarrow I^{\mathrm{f}}_{\mathrm{u,v}} \cdot H\ \forall(\mathrm{u,v}) \in ROI^{\mathrm{f}}_{\mathrm{u,v}}$       // transform camera image
 7:     $\Delta_{\mathrm{i}} = \max(f_{CCF}(J^{\mathrm{i}}_{\mathrm{u}^T,\mathrm{v}^T},\ O^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T}))$       // compare images by CCF
 8: end for
 9: $\mathrm{i}_{\max} \leftarrow \max(\{\Delta\})$                         // find best top view image
10: $J^{\mathrm{f}}_{\mathrm{u}^T,\mathrm{v}^T} = J^{\mathrm{i}_{\max}}_{\mathrm{u}^T,\mathrm{v}^T}$                       // select improved top view image
```

to the IPM full panorama image. This is especially beneficial for the IPM lane line marking annotations, because they are annotated in the IPM full panorama image and taken over into the single top view images, see next subsection 2.2.5.

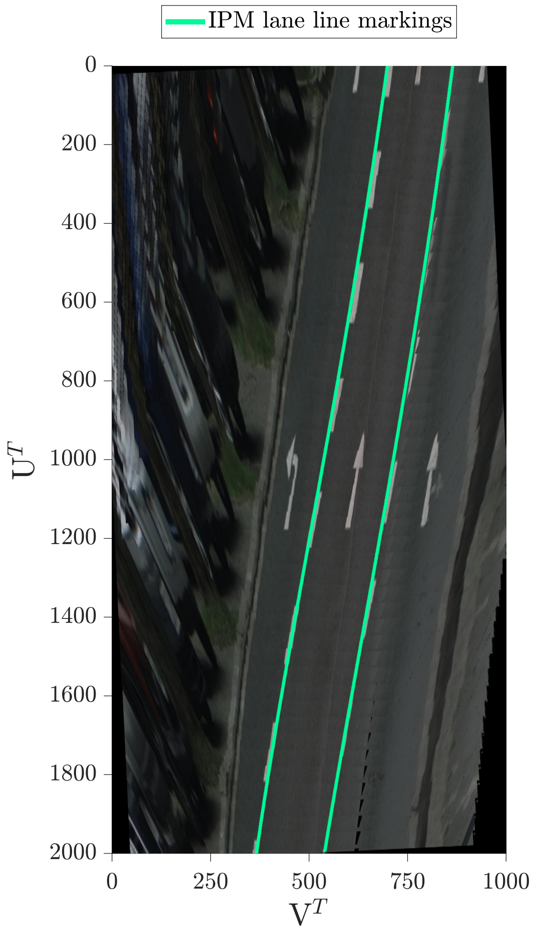
2.2.5 IPM Lane Line Marking Annotations

All IPM full panorama images represent the entire driving route of all intersection scenarios. The idea is to use the IPM full panorama images to annotate all lane line markings with less time investment and adequate quality. The benefit is that only one large IPM full panorama image has to be annotated per intersection scenario instead of many single camera images. In detail, each IPM full panorama image is used for the IPM lane line markings as follows:

All lane line markings are annotated particularly favourably in the IPM full panorama



(a) Top view image.



(b) IPM panorama top view image

Fig. 2.10 Illustration of IPM lane line markings and a comparison between a single top view image and the extracted IPM panorama top view image for the same frame.

image as consistent markings by hand. The area of each associated IPM panorama top view image lane line markings is taken over to its related single top view image. Figure 2.10 (b) displays the annotated IPM lane line markings in the IPM panorama top view image for the ego-vehicle lane, and figure 2.10 (a) displays the accepted IPM lane line markings in the top view image (see light green annotations).

Moreover, the lane line markings in the top view images can be re-transformed into the original camera image by using the inverse transformation matrix H^T for each frame, cf. figure 2.11.

Figures 2.10 (a) and 2.11 also show a qualitative comparison of the example image or frame. The annotated IPM lane line markings are compared to its reference lane line markings for the ego-vehicle lane. In order to check the exactness of the annotated IPM lane line markings, a sample of 60 reference camera images was used. These reference camera images were selected at random. For each reference camera image, the ego-vehicle lane line markings were annotated by hand. These reference lane line markings were re-transformed into their top view images with the explained IPM transformation and compared by their lateral error.

Table 2.1 states the quantitative comparison results between all reference camera images and the top view lane line markings in pixels and in meters for the ego-vehicle lane. One pixel is equal to 0.02 meters in a top view image. The MAE and Root Mean Squared Error (RMSE) are used to measure the exactness in the lateral direction. Table 2.1 shows the calculated errors for all reference camera images, the best, and the worst reference image. Over all reference camera images, an MAE of 13.3 cm is achieved. The maximum difference between the best and worst comparison results is one order of magnitude with an MAE of 3.6 cm to 33.0 cm.

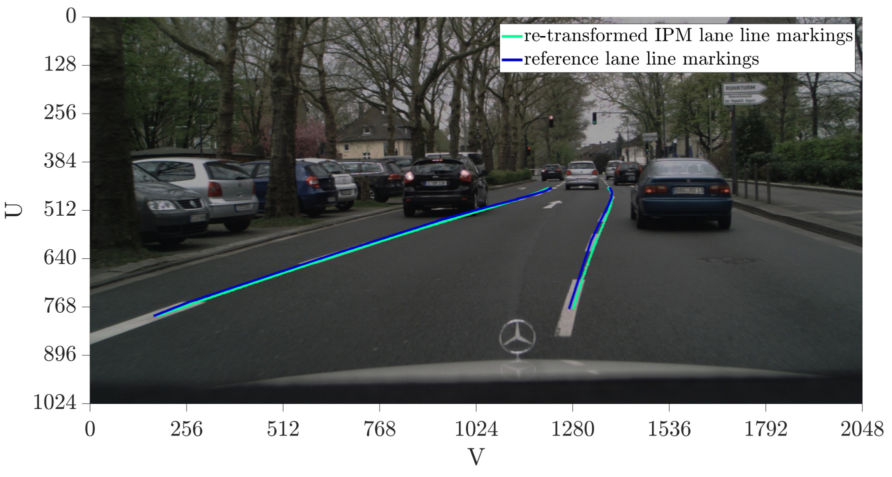

re-transformed IPM lane line markings
reference lane line markings
U
V
0
128
256
384
512
640
768
896
1024
0
256
512
768
1024
1280
1536
1792
2048

Fig. 2.11 IPM lane line markings re-transformed into the camera image and compared to the reference lane line markings for the ego-vehicle lane.

Tab. 2.1 Evaluation of IPM lane line markings by using a sample of 60 references camera images.

References	**MAE**	**RMSE**
All (Mean)	6.66 px / 0.133 m	8.10 px / 0.162 m
Best	1.78 px / 0.036 m	2.07 px / 0.041 m
Worst	16.52 px / 0.330 m	17.51 px / 0.350 m

However, the MAE and RMSE are conditioned by the IPM transformation and depend on the longitudinal distance of the IPM lane line markings, see z(u) in equation 2.24. Figure 2.12 displays the average lateral MAE and RMSE over the longitudinal distance between all reference camera images and IPM lane line markings. It is apparent that the lateral error rises with increasing longitudinal distance. The MAE and RMSE maxima are at a longitudinal distance of 48 m with 26 cm and 34 cm. Moreover, the MAE and RMSE over the longitudinal distance of the IPM lane line markings, which belongs to the qualitative comparison in figure 2.10 (a), is displayed in figure 7.2 in the appendix.

The advantage of this procedure is that the annotation of lane line markings in an IPM full panorama image is carried out much faster than the standard annotation procedure[22] for each single camera image. This means in numbers that all lane line markings are finished annotated on average in about six minutes in an IPM full panorama image or

[22]Normally, camera image for camera image is annotated by hand with an annotation tool that also uses some tracking filters for the lane line markings to relive the annotation.

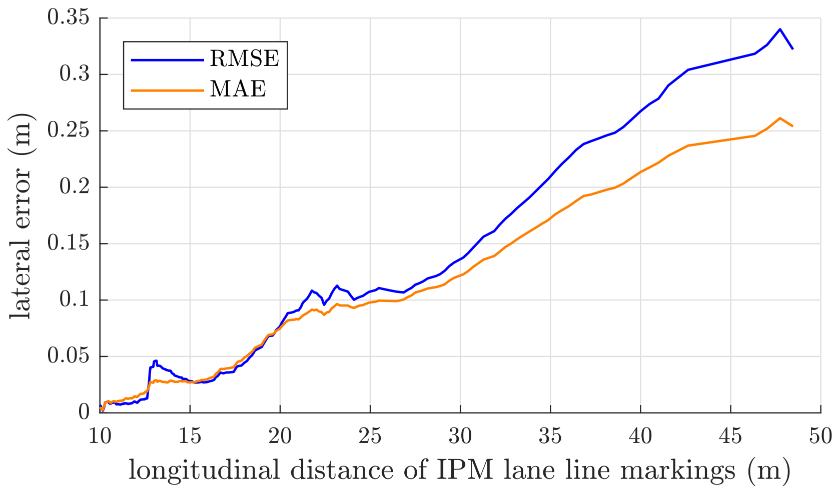


Fig. 2.12 Evaluation of the lateral error over the longitudinal distance of IPM lane line markings. The lateral error is calculated between all IPM lane line markings and re-transformed lane line markings of the reference camera images and displays as MAE an RMSE.

sequence. In contrast, the annotation of a sequence needs approximately 45 minutes[23] on average by use of the standard annotation procedure with an average number of 30 frames to annotate. However, this temporal advantage in the manual annotation outlay is based on sufficient computing power as the generation of an IPM full panorama image takes about two minutes[24].

[23]This number is extrapolated based on the reference lane line markings annotation.
[24]By use of a single 2.4 GHz thread from an Intel Xeon E5-2640 v4 CPU.

3 Manuscript: Automatic Traffic Light to Lane Association for Complex Intersections, IEEE Conference on Intelligent Transportation Systems (IEEE ITS)

Exposition This chapter deals with my "Automatic Traffic Light to Lane Association at Complex Intersections" paper. It was published at the IEEE conference on Intelligent Transportation Systems in 2018. This IEEE ITS paper proposes an approach to resolve the TL2LA problem in the research field of ADAS for the first time in literature. The approach presents a visual solution by using a CNN and focuses on the traffic light to ego-vehicle lane association, which is compared against a test with human subjects named subjective test to evaluate the CNN model performance. The TL2LA problem is methodically resolved by using a target regression vector that transfers the assignment problem to a regression problem. This target regression vector trains a CNN on the column positions of all relevant traffic lights per frame.

This paper refers to the database introduced and uses the traffic light annotations as well as the traffic light to lane assignment ground truth, cf. chapter 1. Applicable are the neural network layers explained (convolutional, pooling, batch normalisation, and fully connected layer), cf. subsection 2.1.3, with all defined hyper parameter employed to successfully train a CNN, cf. subsection 2.1.4. Moreover, reference is made to the IPM method mentioned to transform the road part of the CNN input images into a top view image, cf. section 2.2.

The following contents of this chapter 3 were taken identically from the original publication, see [66], and were adapted to the format of my dissertation only.

Abstract We introduce an approach that enables associating all relevant traffic lights to the ego vehicle lane at complex intersection scenarios. To achieve this, we use a combination of image data and traffic light metadata.

First, we prepare the input image data that consists of a partly Inverse Perspective Mapping (IPM) for the road and highlight traffic light metadata information like position and colour in the image. Next, we train with the input data a common Convolutional Neural Network (CNN) topology on a target regression vector. This vector contains the column positions of all relevant traffic lights in the image. Finally, we perform two post-processing steps that improve single traffic light results.

Results on average yield an accuracy of 86.40 %. In the near field, accuracy improves to 93.88 %. Moreover, we carry out a subjective test, where humans are presented with the same images as processed by the CNN. This defines a reliable baseline for detailed comparisons against our results.

In sum, our approach and the subjective test achieve similar results on average. In detail, the subjective test reaches more accuracy for highly complex intersection scenarios. However, it has less accuracy for distances between 25 m to 50 m to the stop line than the proposed approach.

3.1 Introduction

The association of all relevant traffic lights to the ego vehicle lane represents a field which did not receive a lot of attention so far. However, performing traffic light association is a fundamental step in Traffic Light Assistance Systems (TLA) and provides an essential component for autonomous level three and higher driving in cities.

The problem is presented in fig. 3.1. The generic intersection scenario consists of five lanes, six traffic lights, and all correctly assigned relevant traffic lights to the ego vehicle lane number four. For some cases, not all traffic lights are a decision factor, which are identified with the ego vehicle lane borders, see traffic light four in fig. 3.1. More generally, we try to solve a one-to-many assignment problem with one lane and many traffic lights.

One TLA application regarding autonomous driving consists of braking and stopping on lines at signalled intersections. Therefore, the application requires detailed information about the traffic lights that are relevant for the ego vehicle lane. The main challenges to produce this detailed information are: (i) all the relevant traffic lights are rarely positioned above or beside their respective lanes, (ii) some traffic lights are positioned close together but are not relevant for the same lane, (iii) traffic lights that are not relevant have to be rejected, i.e. tram, pedestrian and bicycle lights as well as traffic lights, which belong to the next upcoming or neighbour intersection.

The proposed approach delivers all relevant traffic lights without prior knowledge while approaching the intersection based on a camera sensor. The approach is independent of any cloud systems and statistical observations over-time of equal intersections. We use

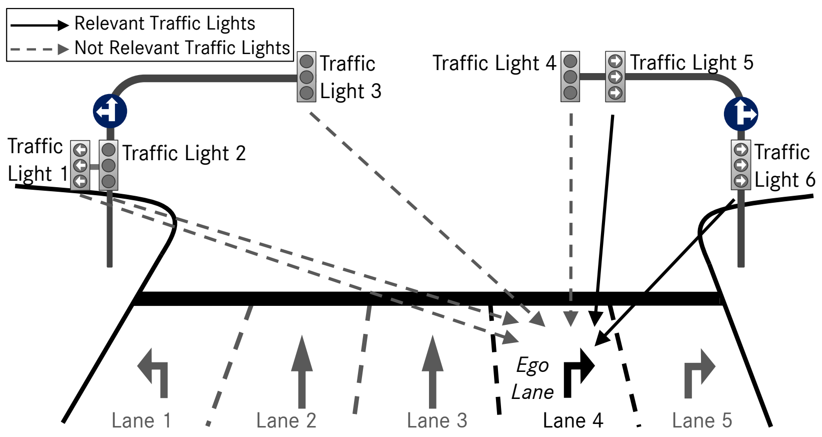


Fig. 3.1 Generic example to associate all relevant traffic lights to the ego vehicle lane.

image data with annotated traffic lights and allocations to the ego vehicle lane. This data is used to train a Convolutional Neural Network (CNN). We use annotated traffic lights, because the problem to detect traffic lights has already been addressed by many previous work in the research field of Traffic light Detection (TLD), see next sec. 3.2. The CNN delivers a vision-based solution for the association of all relevant traffic lights to the ego vehicle lane.

In this paper, we will not focus on the generation of the annotated traffic lights, because this is part of other research fields like TLD. We use daylight image data with sunny or cloudy weather conditions.

This paper is structured into four main parts. First, we put our recent work in context to related research fields, sec. 3.2. Second, we introduce our CNN model approach, sec. 3.3. Third, we explain our dataset and compare the results that are observed by our CNN approach to two simpler rule-based approaches. Finally, the conducted subjective test ensures a discussion and a meaningful evaluation, sec. 3.4

3.2 Related Work

Our paper topic has two related and independent research fields. One research field is TLD, which is a necessary condition to solve our presented problem. One of the recently published TLD approaches is [21]. They introduced three methods for using stereo vision to detect traffic lights. These methods improve the false positive rates while minimally decreasing detection rates. Another method was presented in [16]. They build a vision based detection method by combining Histogram of Oriented Gradients features with a linear Support Vector Machine (SVM) to detect traffic lights plus a CNN to classify traffic lights. Further methods to detect traffic lights are the use of

Adaptive Background Suppression Filter together with a SVM [17] or pure CNN based methods in [22] and [23].

The other research field is TLA, which needs reliable information about the relevance of traffic lights. One TLA approach is to support the driver with additional information about the next upcoming traffic light, e.g. colour and distance. In [67] and [68] they made approaches with interconnected traffic lights to the ego vehicle. They are based on interface to vehicle communication and GPS positions of all subscribers. Both approaches work with a connection to a global database, which provides among others the relevant traffic light information. Another TLA approach is [69], which focuses more on safety during autonomous driving. They made a simulation controlled by traffic lights whether a vehicle should stop or could drive over the intersection. This could be one of the most common TLA systems in future.

The presented problem to associate all relevant traffic lights has little of its own specific research literature yet. However, it is sometimes mentioned in publications which deal with TLD. In [70] an overview of traffic light research was presented, focusing on available datasets and unsolved problems in the context of TLD. They also pointed out that a Driving Assistance System (DAS) must be able to determinate whether a traffic light is relevant or not relevant for the ego vehicle lane at intersection scenarios. The first approach for this is a static one in [71]. They recorded a database before their test drive, in which the information about relevance is stored for each traffic light on the route. A more dynamic approach is the traffic light mapping [72]. They determine the relevance by assuming that the largest and nearest orientated traffic light in driving direction is valid based on the intersection width and the estimated real world positions of traffic lights. In addition, their hypothesis is statistically improved by traversing the same intersections many times. The newest approach in literature was made in [73]. They defined their so-called main traffic light. This traffic light has to be detected for DAS at multi traffic light intersections, which satisfies the same purpose as one relevant traffic light. Their main traffic light is defined as follows: it has to be the traffic light with the largest size and highest position from the group with most traffic lights of the same colour. Unfortunately, all of these approaches have not yet performed any experiments that provide results for direct comparisons regarding the association of relevant traffic lights.

Our main contribution is an approach to associate all relevant traffic lights to the ego vehicle lane based on a camera sensor during a single intersection crossing. We develop a CNN, which is fed with specially processed image data and therein encoded metadata with characteristics of the traffic lights. The CNN is trained on a target regression vector, which contains all positions of relevant traffic lights.

We compare this novel approach to the two previously mentioned rule-based approaches, the traffic light mapping [72] and the main traffic light approach [73]. Furthermore we evaluate our CNN results with a subjective test by showing humans our CNN test images and request them to associate the relevant traffic lights.

3.3 Proposed Approach

For our work we use data from the DriveU dataset [12] recorded with the left-sided stereo camera. From this dataset, we selected exclusively complex urban intersection scenarios. We define a complex intersection scenario in terms of two conditions. First, a decision regarding the relevance of traffic lights must always be made. Second, at least one of three main challenges, mentioned in sec. 3.1, must exist.

Our proposed CNN model approach consists of four working steps, cf. fig. 3.2. The steps are depicted as a flowchart. In the first step, we prepare the input images and target regression vectors from the frames and traffic light metadata that ensures better understanding by the CNN. In the second step, the CNN is set and trained by the input and target data. Next, we perform an output vector mapping to examine single traffic light results. Last, we execute a simple majority decision maker over each sequence by its temporally sorted frames to improve the results.

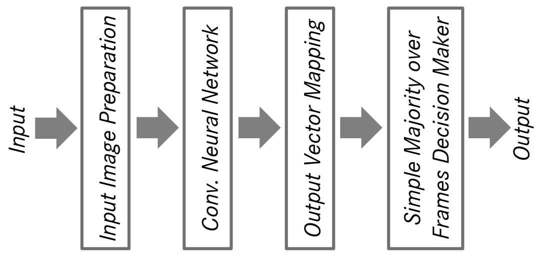


Fig. 3.2 Flowchart of our proposed CNN model approach.

3.3.1 Input Image Data Preparation

The input images for our CNN model consist of three different down sampled and square tailored RGB colour planes from the original image. These planes yield superimposed an input image of size (256, 256, 3) pixel, cf. fig. 3.3 (a).

Plane 1 contains the traffic light metadata information about position, size, and colour. The information of traffic lights is processed such that each traffic light's height and width is three-times increased. The rectangles are filled with colours that group traffic lights which have the same information. We increase the area of a traffic light position in the plane, because this increases importance of the traffic light information. Otherwise, the traffic light size is too small and the CNN is not able to react to it.

Additionally, we place the original traffic light appearance with a zoom factor of two in the centre, thus we do not lose any information like traffic light arrows or icons. Plane 2 contains the originally recorded camera image, but traffic lights from plane 1 have been cut out. We restricted the image up to a maximum distance that is the next 50 meters of the road. If the distance to the stop line of the intersection is smaller than 50 meters, we will restricted the image until the stop line. Plane 3 contains the missing road meters from plane 2. These parts are transformed into an Inverse Perspective Mapping (IPM) by using the extrinsic and intrinsic camera parameters together with a flat world assumption, cf. [61]. In consequence, the input images become robust to asymmetric shape transformations of lane markings, arrows, and icons between different input images. These modifications enable the CNN to be more easily trained on the lane features, see sec. 3.4.3 where we used different input image variants.

In Fig. 3.3 (b) three examples of generated CNN input images are presented. The perspective of the ego vehicle lane is always identified in the vertical centre of the input image. This results from the fact that the ego vehicle is always in the lower centre of the image because of the camera alignment.

The target data are target regression vectors that have the length of one input image row. In fig. 3.3 (c) the corresponding target regression vectors are shown for the three example input images from (b). In the original data, the traffic lights do not contain

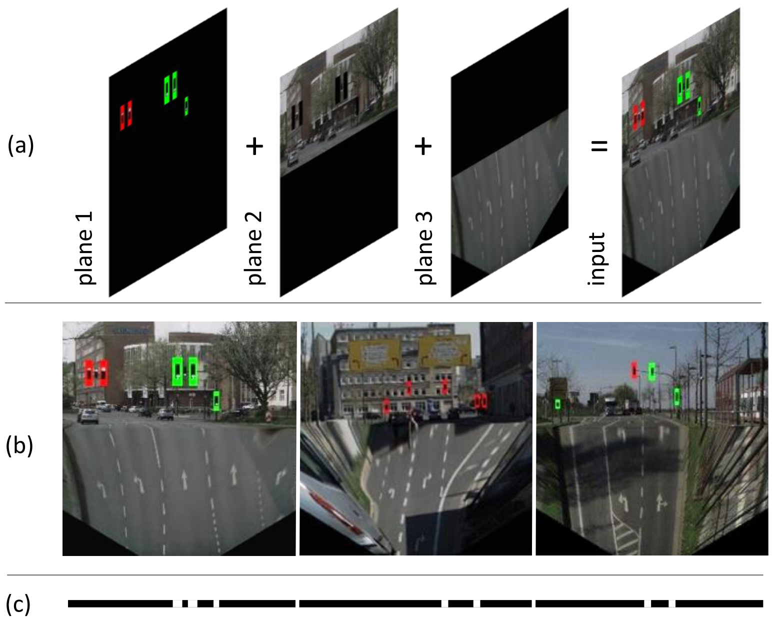


Fig. 3.3 (a) input image preparation: plane 1 with traffic lights of intersection, plane 2 with upper camera image, and plane 3 with bottom IPM image, (b) three input image examples, (c) target regression vectors corresponding to input image examples.

different information within a column. This allows us to train and test the relevant and not relevant traffic lights according to their column positions. If a traffic light is relevant for the ego vehicle lane, we set the target regression vector at this position to 255 or white respectively. Otherwise, we set the target regression vector to zero or black respectively. The white gaps in fig. 3.3 (c) have the same width as the enlarged relevant traffic lights.

3.3.2 Convolutional Neural Network

The CNN topology possess five convolutional (Conv.) and maximum pooling layers (Max. Pooling). The down sample factor is symmetric and has a factor of two due to the max pooling stride parameter of 2 by 2. As a kernel size parameter, in all convolutional layers a 3 by 3 shared weight kernel is used. In other work, the choice of small kernel and stride parameters has proven to be particularly effective [39].

The last maximum pooling layer is followed by two fully connected layers (FC.). The output layer is the output regression vector. As activation function, we apply the rectified linear unit function to all layers. Figure 3.4 presents the described CNN topology and contains the number and size of each feature map. Moreover, after each convolutional layer, a batch normalisation layer that decreases the training time is added. Dropout regularization is applied to the two FC. layers to avoid over-fitting during the training of the CNN.

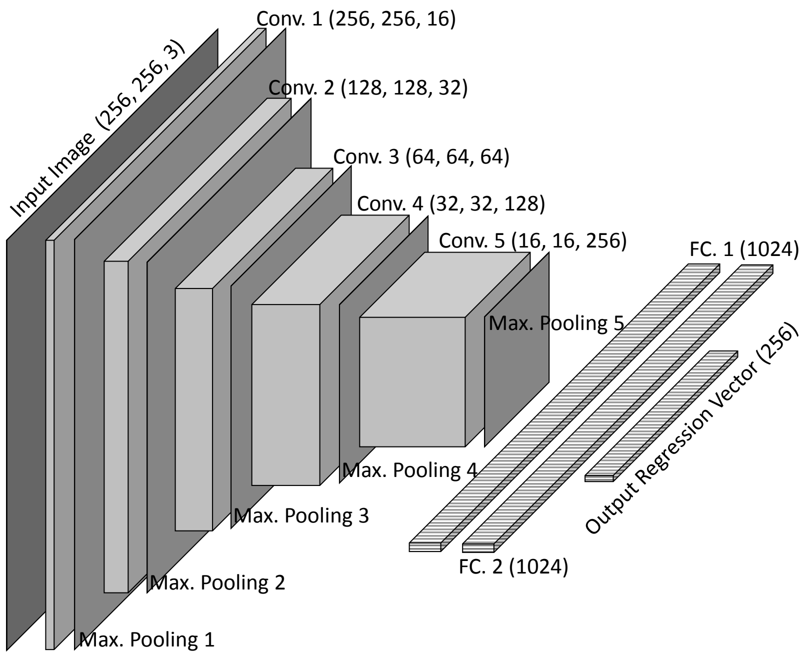


Fig. 3.4 CNN topology for the traffic light relevant association.

The CNN is trained with a decreasing learning rate of factor 1.25. It begins at every 25 epochs with an experimentally determined initial learning rate of $5 \cdot 10^{-6}$. We deploy as loss function the mean squared error function. The training algorithm utilizes the stochastic gradient descent momentum. The training is stopped if the validation error stagnates or increases for five successive epochs. The maximum number of possible epochs is set to 125, and the mini batch size amounts to 200 frames per iteration.

3.3.3 Output Vector Mapping

As CNN output we get a regression vector, which contains the traffic light relevant information coded through its column positions. Since we want to evaluate single traffic light results, we have to perform an output vector mapping as post-processing on the test dataset. Therefore the associated relevant and not relevant traffic lights in the output regression vector are mapped out of the frame to single traffic lights for each sequence.

We need three vectors for the mapping: the target regression vector tv, the output regression vector ov and a matching vector mv. The matching vector contains the information regarding the traffic light identifier t per sequence. We are able to generate the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) confusion matrix values for each single traffic light with these vectors. Therefore we use an ascertained length l_t as well as a left side starting position p_t from the matching vector for each single traffic light, see fig 3.5. We compare a threshold $th_{128} = 128$ against the sum of every chosen element i of the output regression vector. With this threshold decision, we determinate the traffic light relevant result r for each traffic light t in a frame f of a sequence s:

$$r_{\mathrm{s,f,t}} = \left(\frac{1}{l_\mathrm{t}} \sum_{\mathrm{i}=p_\mathrm{t}}^{l_\mathrm{t}-1} ov_{\mathrm{s,f}}(\mathrm{i}) \geq th_{128} \mapsto 1; 0 \right) \tag{3.1}$$

In the traffic light relevant result $r_{\mathrm{s,f,t}}$, a value of one means a relevant traffic light, and a value of zero means a non-relevant traffic light. The confusion matrix values $c_{\mathrm{s,f,t}}$ are generated through, cf. fig 3.5,

$$c_{\mathrm{s,f,t}} = \left(r_{\mathrm{s,f,t}} = 1 \mapsto \left(\sum_{\mathrm{i}=p_\mathrm{t}}^{l_\mathrm{t}-1} tv_{\mathrm{s,f}}(\mathrm{i}) = 0 \mapsto \mathrm{FP}; \mathrm{TP} \right) \right) \tag{3.2}$$

or

$$c_{\mathrm{s,f,t}} = \left(r_{\mathrm{s,f,t}} = 0 \mapsto \left(\sum_{\mathrm{i}=p_\mathrm{t}}^{l_\mathrm{t}-1} tv_{\mathrm{s,f}}(\mathrm{i}) = 0 \mapsto \mathrm{FN}; \mathrm{TN} \right) \right). \tag{3.3}$$

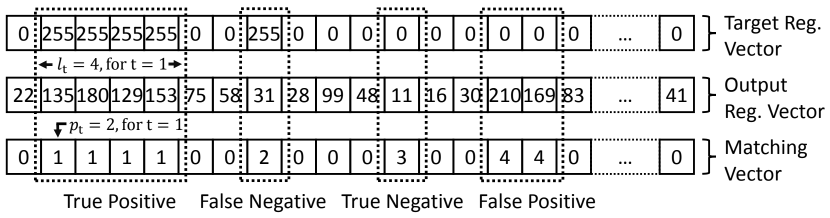


Fig. 3.5 Examples for TP, TN, FP and FN relevant traffic lights generated from the target regression, output regression and matching vector for one frame of a sequence.

3.3.4 Simple Majority over Frames Decision Maker

We employ a simple majority over frames decision maker on all results $r_{\mathrm{s,f,t}}$, cf. equation 3.1. Hence, we reduce the number of flipping outputs for the same traffic light between consecutive frames of a sequence. We always calculate the mean traffic light relevant results from the first to the current frame of a sequence. Then we decide by simple majority whether it is a relevant or a not relevant traffic light to improve the $r_{\mathrm{s,f,t}}$ result. This is achieved by iterating over s and t and forming the mean value by add up all previous frames k until the current frame f:

$$r_{\mathrm{s,f,t}} = \left(\sum_{\mathrm{k=1}}^{\mathrm{f}} r_{\mathrm{s,k,t}} > \frac{\mathrm{f}}{2} \mapsto 1; 0 \right) \tag{3.4}$$

In case of $\sum_{\mathrm{k=1}}^{\mathrm{f}} r_{\mathrm{s,k,t}} = \frac{\mathrm{f}}{2}$ we decide in favor of the newest value $r_{\mathrm{s,f,t}}$. The final confusion matrix values $c_{\mathrm{s,f,t}}$ are calculated with the explained improvement as shown in equation 3.2 and 3.3.

3.4 Experiments

3.4.1 The Dataset

We use for our experiments the data illustrated in table 3.1. We split the data to a training (≈89 %) and test (≈11 %) dataset. Each sequence represents a drive recorded with a camera to an intersection from different distances until the stop line is reached by the ego vehicle. We extracted from each sequence 25 to 30 frames. All frames contain annotated traffic lights, which are marked as relevant or not relevant for the ego vehicle lane. In sum we have a majority event rate (share of relevant traffic lights) of 46.7 % in the dataset.

The training dataset is balanced with multiple copies of underrepresented intersection scenarios, because we have, e.g. more one-left-two-straight-lane scenarios than one-left-one-straight-one-right-lane scenarios in the dataset. Hence, we avoid over-training

Tab. 3.1 Dataset overview with number of sequences, frames, and all traffic lights of the training and test dataset.

	Sequences	**Frames**	**Traffic Lights**
Training Dataset	356	10360 (64082)[a]	49363 (308312)[a]
Test Dataset	48	1310	7136
Sum	404	11670	56499

[a] With data balancing and augmentation.

of frequently existing scenarios. Moreover we use data augmentation for the training dataset by doubling each frame and swapping the traffic light colours from red to green or the other way around. This prevents over-training on a specific colour for a relevant traffic light through equal distribution of all red and green traffic lights. Thereby we increase the number of frames and traffic lights in the training dataset sixfold, cf. table 3.1.

Figure 3.6 shows a histogram of all available frames in the test dataset against the distance d in meters to the stop line for all sequences. The most interesting distance is between 15 and 45 meters, where the ego vehicle is close enough to see some traffic light icons like arrows and far enough away to have even the higher traffic lights visible in the frame. We divided our test dataset into four distance ranges for an easier evaluation: $d < 12.5$ m, $12.5 \text{ m} \leq d < 25.0$ m, $25.0 \text{ m} \leq d < 50.0$ m, and $d \geq 50.0$ m.

Additionally, we divided our test dataset into four lane groups derived from the number of lanes L: $L = 2$, $L = 3$, $L = 4$, and $L \geq 5$. Table 3.2 displays the distribution for the number of sequences against the lane groups of the test and training dataset.

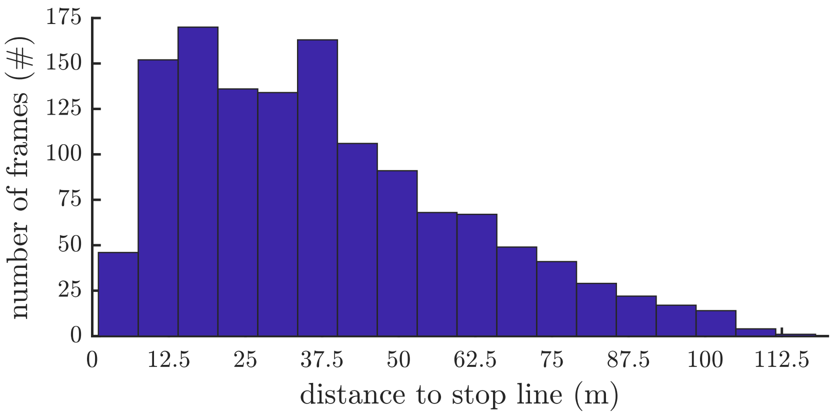


Fig. 3.6 Dataset histogram for number of available frames against distance to stop line at intersection of the test dataset.

Tab. 3.2 Dataset distribution for number of sequences against number of lanes L at intersection of the test and training dataset.

	$L = 2$	$L = 3$	$L = 4$	$L \geq 5$
Test Sequences	3	13	20	12
Training Sequences	22	86	169	79

3.4.2 Comparison with rule-based Approaches

We have ourselves implemented the traffic light mapping approach from [72] and the main traffic light approach from [73], mentioned in sec. 3.2. However, the traffic light mapping approach is implemented without the statistical improvement through many observations of the same intersections. These we compare against our CNN model approach.

In contrast to our CNN model, both approaches select only one traffic light per frame and mark this as relevant. Hence, we have to align our approach to the other ones. Therefore we use, on the one hand, precision as evaluation metric, because the rule-based approaches do not care about FN or TN relevant traffic lights. The precision is defined as

$$precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{3.5}$$

On the other hand, we select only one relevant traffic light per frame, too. In our case, we select the traffic light $\mathrm{t_{max}}$ with the highest confidence from all as relevant predicted traffic lights T.

$$r_{\mathrm{s,f,t_{max}}} \Leftarrow \max(\{C_{\mathrm{s,f,t}}(\{r|1\}) : \mathrm{t} \in T\}) \tag{3.6}$$

The single confidence per traffic light, frame and sequence is calculated as follow in terms of the output regression vector

$$C_{\mathrm{s,f,t}} = \frac{1}{l_{\mathrm{t}} \cdot th_{128}} \sum_{\mathrm{i}=p_{\mathrm{t}}}^{l_{\mathrm{t}}-1} ov_{\mathrm{s,f}}(\mathrm{i}). \tag{3.7}$$

Hence, we ensure a correct comparison between their and our approaches.

Table 3.3 exhibits the precision results for both rule-based approaches and our CNN model approach generated on our test dataset. The results yield an up to 25 % lower precision than our approach. This underlines that simpler rule-based approaches do not work for the complex intersection scenarios which occur exclusively in the dataset used.

Tab. 3.3 Comparison between our CNN and two simpler rule-based approaches on the test dataset by mean of precision.

Approaches	**Precision**
Traffic Light Mapping	62.37 %
Main Traffic Light	71.18 %
Our CNN Model	87.93 %

3.4.3 Different Variants of Input Images

We tested our approach against different input image variants, too. Figure 3.7 (a) shows the used variants: image (i) is based on the background only, image (ii) contains the background and the recorded road by the camera, and image (iii) is the preferred image variant for our approach with the IPM transformed road. Figure 3.7 (b) displays the results over distance to the stop line of the three input image variants. Thereby σ indicates the standard deviation of several trained and tested CNNs.

On the one hand, it appears that the variants (ii) and (iii) perform generally (except in the near field) significantly better than variant (i), which does not have any road features

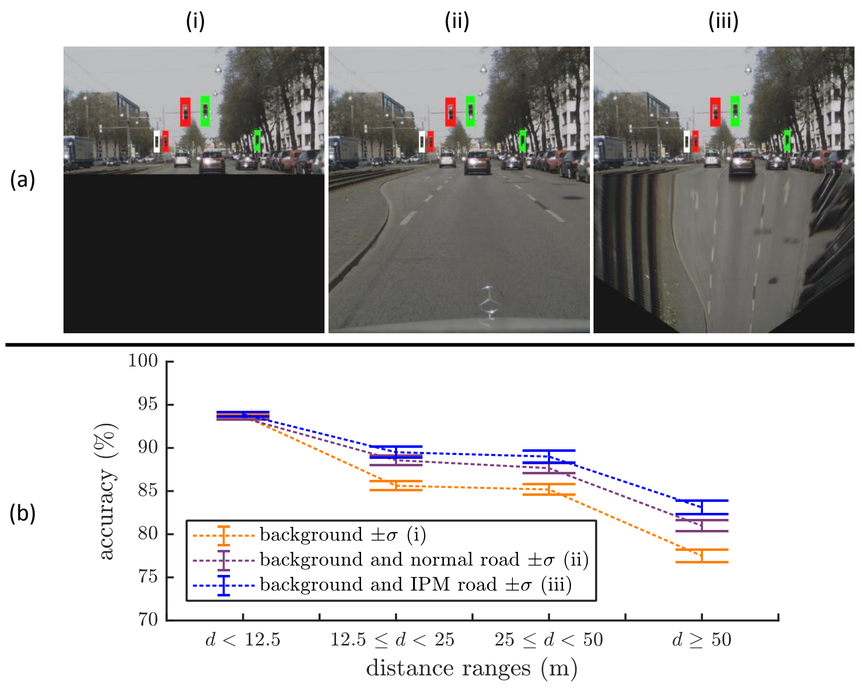


Fig. 3.7 (a) three input image variants from (i) to (iii) and (b) results over distance ranges of the three input image variants from (a).

in the image. On the other hand, variant (iii) achieves for longer distances ($d > 25$ m) significantly more accuracy than variant (ii), because it is using IPM transformed road features, cf. fig. 3.7.

3.4.4 Subjective Test on CNN Image Data

We conducted a subjective test with the same input images that we use for our proposed CNN model approach to define a reliable baseline, cf. fig. 3.7 (a) (iii). This baseline is used for a meaningful comparison to our results.

For humans, traffic light to lane association is a continuous process that happens when approaching an intersection. In contrast to this and to provide a fair comparison with our automatic method, we decided to chunk the human recognition process by asking our subjects to tell us their traffic light to lane association using single images of the same intersection taken at four different distances. For this, we used a ten seconds recognition time window in order to allow more than the usual time to assess each single image. Usually one would calculate about 5 to 6 seconds of driving time up to any traffic light starting from a distance of about 75 m. Hence, in this way we are essentially measuring an upper assessment bound for single image-based recognition by humans (note, longer times did not lead to better results).

In detail, as test data we extracted from each of the 48 test sequences one frame per available distance range. Each frame was randomly selected for each distance range. This reduced the amount of test data frames for the subjective test. In sum we selected 100 test images for the subjects plus ten practice images. The practice images are given to the subjects to get familiar with the valuation tool before the actual test starts. All test and practice frames were upsampled from size (256, 256, 3) to (512, 512, 3) pixels using nearest neighbour interpolation to obtain a user-friendlier eye-hand coordination on the screen. The training images were not shown to the subjects by us, because on the one hand the training dataset has a huge number of images. And on the other hand we assume that the subjects have sufficient experience in urban traffic. An overview of the chosen subjective test dataset taken out of the testing dataset is shown in table 3.4.

During the practice and test stage all subjects have ten seconds per image time to

Tab. 3.4 Overview of the subjective test dataset as random selected subset of the full test dataset.

	Sequences	**Frames**	**Traffic Lights**
Subjective Test Dataset	48	100 (110)[c]	520

[c] Including ten practice images.

associate all relevant traffic lights to the ego lane. If the time per image is over, the screen will switch to black and no further changes are possible. This is done because the subjects should not start analysing the displayed scenarios in detail. In reality they also have a limited time to decide. The subjects can mark a traffic light as relevant by directly clicking on the concerned traffic light in the image.

We recruited 21 subjects (20 male and 1 female) with a mean age of 30.2 years and a mean driving experience of 10.4 years. All subjects had essentially no knowledge of the shown intersection scenarios in the test images, which were recorded in the German cities of Essen, Duesseldorf, Dortmund and Hannover. The mean decision time over all subjects per test images was 5.97 seconds, thus the processing time of ten seconds was almost always adequate. The maximum processing time was not longer than 20 minutes, which excluded errors due to fatigue of the subjects.

3.4.5 Results

We evaluated our results by means of average over distance to the stop line and over the number of lanes. In addition, we compared them against the subjective test results. We make use of the metric accuracy, which is defined as

$$accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}. \tag{3.8}$$

Table 3.5 contains the average accuracy for our CNN model test dataset and the subjective test dataset as well as the 90 % confidence interval of the subjective test estimated from our 21 subjects. The results are close together with an absolute difference of 0.87 % accuracy. The difference between the average accuracy for our CNN model on the subjective and full test dataset is 0.39 %. This low value indicates that the subjective test dataset is a representative sample of the full CNN model test dataset.

Figure 3.8 compares the results for each defined distance range. The subjective test results increase gradually when getting closer to the stop line as we would also expect more distinctly from the CNN model. But our CNN model achieves in the distance

Tab. 3.5 Average accuracy for our CNN model and the subjective test of the subjective test dataset.

	Average Accuracy	**90 % Confidence Interval**
Our CNN Model	86.79 (86.40)[d] %	—
Subjective Test	85.97 %	±1.00 %

[d] Average accuracy of the complete test dataset.

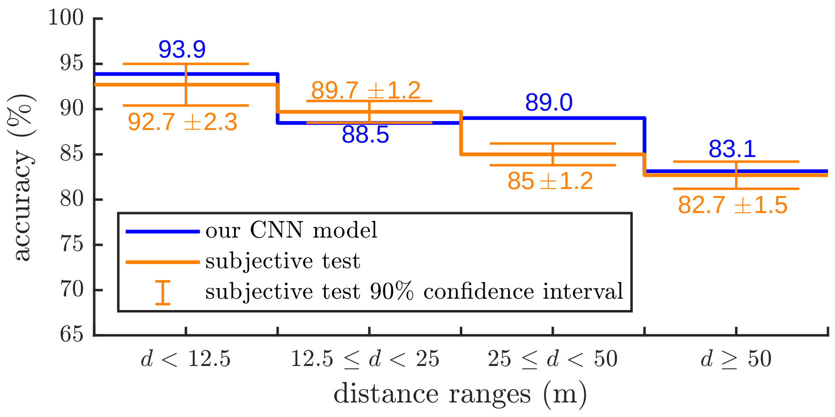


Fig. 3.8 Comparison between our CNN model and the subjective test results over distance ranges deduced from the distance d.

range from 25 m to 50 m about 4.0 % more accuracy than the subjects. This is quite impressive and helpful to associate all relevant traffic lights as early as possible. All other distance ranges do not have significant differences in consideration of their 90 % confidence intervals.

A second comparison (fig. 3.9) illustrates the accuracy for each lane group determined by the number of lanes of each sequence. In two lane intersections, our CNN model reaches about 5.6 % more accuracy than the subjects. For three and four lane intersection scenarios significant differences do not exist. In most complex intersection scenarios with five or more lanes, the subjective test achieves about 4.9 % more accuracy. An explanation for this is that the used dataset probably does not have enough scenarios of this kind to understand any very complex situation, whereas human beings can.

Moreover we have compiled some visual results taken from the subjective test dataset in fig. 3.10 to force a direct confrontation between our CNN model and the subjective

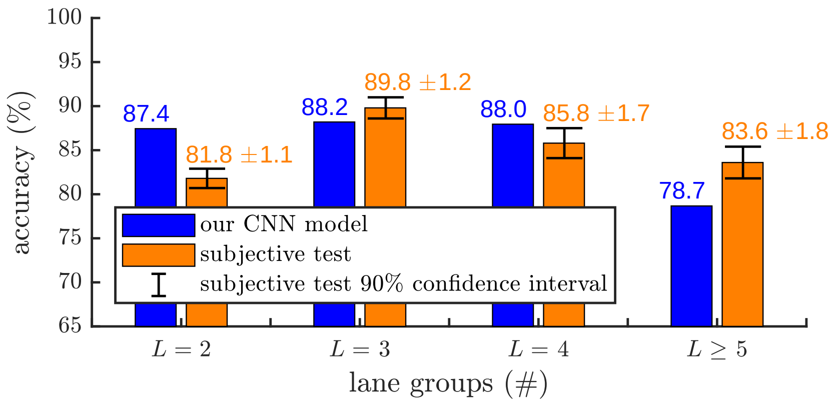


Fig. 3.9 Comparison between our CNN model and the subjective test results over lanes groups deduced from the number of lanes L.

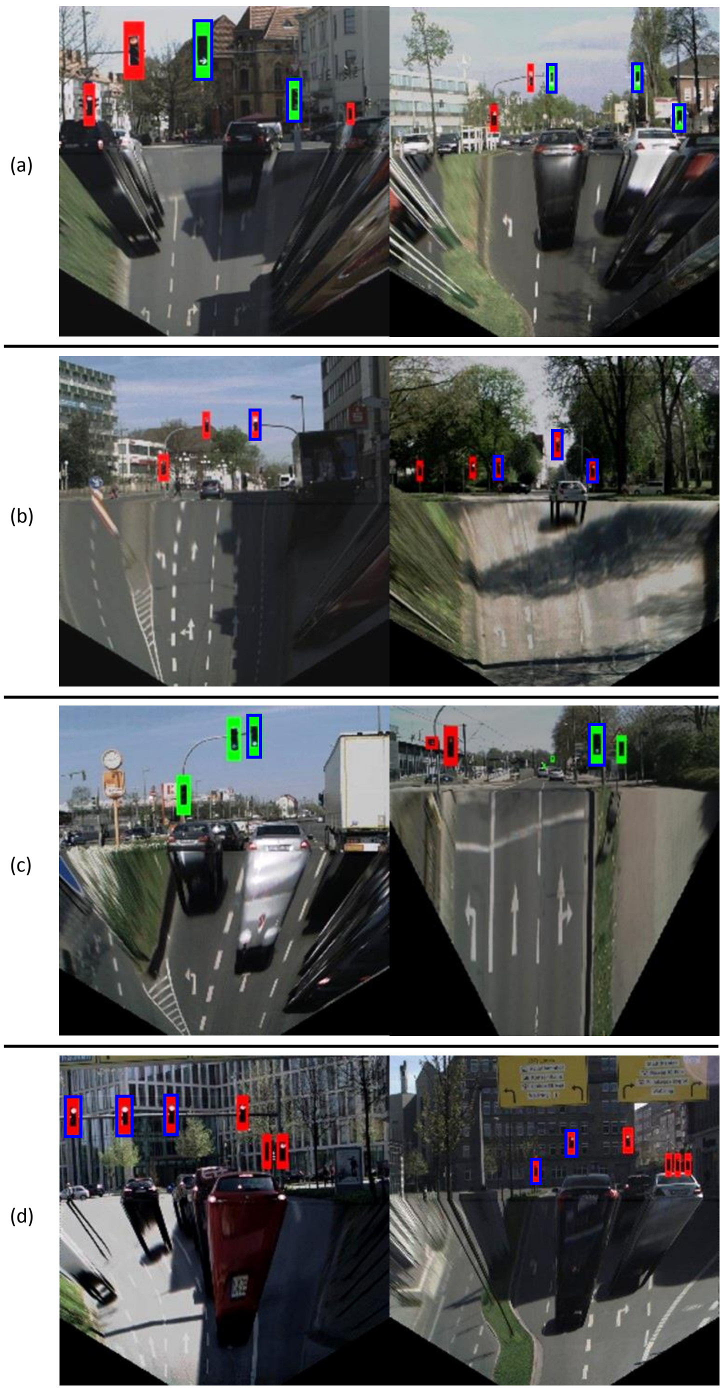


Fig. 3.10 Comparison of our CNN model and the subjective test by single frame results (the blue rectangle boxes visualise the ground truth for relevant traffic lights): (a) CNN model and subjective test associate all relevant traffic lights correctly, (b) CNN model associate all relevant traffic lights correctly and subjective test makes mistakes, (c) CNN model produces errors and subjective test associate all relevant traffic lights correctly, (d) CNN model produces errors and subjective test makes mistakes.

test. The confrontation consists of all four possible comparisons to each other. Each is exemplified with two input images, in which all relevant traffic lights (annotated ground truth) are marked with a blue rectangle box. The first comparison in fig. 3.10 (a) shows two input images, which are completely correctly predicted by both our CNN model and the subjects. The second gives two examples of input images, which are only correctly predicted by our CNN model. These scenarios are particularly demanding because all traffic light colours are identical despite different lane assignments for left and straight lanes, see fig. 3.10 (b). The third comparison displays input images, which are predicted correctly by the subjects only, see fig. 3.10 (c). There is the challenge that traffic lights with different lane assignments hang extremely close together and even have the same colour. The last comparison in fig. 3.10 (d) shows very challenging five lane and four lane intersection scenarios, which are predicted from our CNN model and the subjects with errors.

3.5 Conclusion

In our work we dealt with the problem of associating all relevant traffic lights for the ego vehicle lane. We introduced an approach based on a CNN, which processes traffic light metadata encoded in partly IPM transformed image data as input images. The CNN was trained on a target regression vector, which contains all column positions of relevant traffic lights. The approach was trained and tested with a difficult dataset with more than 400 complex intersection scenarios.

In our evaluation, we compared our approach to two rule-based approaches on the test dataset. This comparison leads to the insight that rule-based approaches are unsuitable for complex intersection scenarios.

In addition, we carried out a subjective test with randomly selected test images to create a baseline, which we used for statistical and visual comparisons. It turned out that our CNN model approach with 86.79 % to 85.97 $\pm$1.00 % is almost at par with the subjective test. However in a detailed evaluation, our approach gave better performance in the distance range from 25 m to 50 m. The test subjects achieved better performance for complex intersection scenarios with five or more lanes. The visual confrontation of single frames resulted in two insights valid for our CNN model approach and the subjective test. The first is that intersection scenarios were predicted fully correctly, where relevant and not relevant traffic lights are differentiable by different colours. The second is that intersection scenarios were predicted with errors, which have four or more lanes and every traffic light shines in the same colour.

We developed our CNN implementation by use of the open-source framework Keras with the TensorFlow backend [43]. The CNN processing time was 1.69 ms per frame with two Nvidia GeForce GTX 1080 Ti.

In the future we will work on additional optimisations of our CNN, e.g. with a pre-trained CNN or different image resolutions.

Acknowledgement We would like to thank Andreas Fregin for providing us with a part of his recorded DriveU dataset and all colleagues and friends, who have conducted the subjective test for us.

3.6 Complementary Investigations to the IEEE ITS Paper

In this section, additional experiments are quoted which were not part of the original paper because of reasons pertaining to relevance and limited space. The chosen CNN topology, input image size, and dataset size are investigated, and the mentioned three traffic light to lane association main challenges are evaluated in detail and concluded.

3.6.1 Investigation of CNN Topology vs. Input Image Size

I investigated the CNN topology and input image size with a comparative simulation study. This simulation study was executed with three different CNN topologies and four various input image sizes for each combination.

The three CNN topologies are illustrated in detail in figure 3.11. The figure shows the different convolutional layers of the three explained topologies: AlexNet (the proposed and used one), VGG-16, and VGG-19. Note that the number of convolutional feature

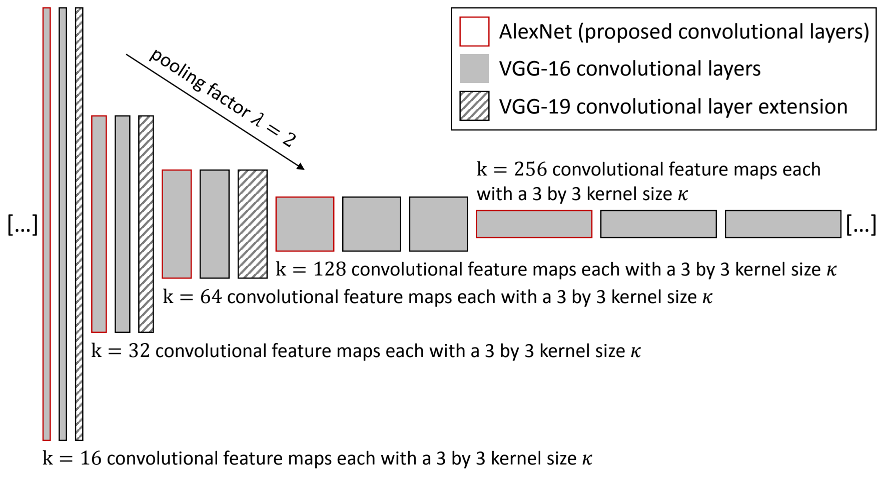

AlexNet (proposed convolutional layers)
VGG-16 convolutional layers
VGG-19 convolutional layer extension
pooling factor λ = 2
[...]
k = 256 convolutional feature maps each with a 3 by 3 kernel size κ
[...]
k = 128 convolutional feature maps each with a 3 by 3 kernel size κ
k = 64 convolutional feature maps each with a 3 by 3 kernel size κ
k = 32 convolutional feature maps each with a 3 by 3 kernel size κ
k = 16 convolutional feature maps each with a 3 by 3 kernel size κ

Fig. 3.11 Convolutional layers of the AlexNet, VGG-16, and VGG-19 topologies. The square brackets ([...]) symbolise the omitted input image layer at the beginning as well as the omitted fully connected layers and the output regression vector at the end of the CNN topology, cf. figure 3.4 in subsection 3.3.2.

Tab. 3.6 Results of the CNN topology and input image size investigation by using the accuracy metric and their 90 % confidence intervals for the IEEE ITS dataset.

	Input Image Size				
Topology	**64 px**	**128 px**	**256 px**	**512 px**	**Mean**
AlexNet	68.2 ±2.3 %	82.4 ±1.4 %	86.4 ±0.5 %	84.7 ±1.0 %	80.4 ±3.2 %
VGG-16	83.9 ±0.6 %	85.6 ±0.6 %	84.5 ±1.5 %	86.8 ±0.8 %	85.2 ±0.7 %
VGG-19	65.4 ±7.1 %	82.0 ±4.3 %	85.7 ±0.4 %	81.0 ±7.2 %	78.5 ±5.5 %
Mean	72.5 ±4.3 %	83.3 ±1.6 %	85.6 ±0.6 %	84.2 ±2.5 %	—

maps k and the pooling factor λ were retained for all topologies and that figure 3.11 shows exclusively the convolutional layer part and not the complete CNN topology.

The AlexNet [29] topology is equivalent to the proposed CNN topology with five convolutional layers and three fully connected layers, but with consistently smaller shared weight kernel sizes of 3 by 3 weights, cf. subsection 3.3.2. It is a comparatively small one in comparison to other popular CNN topologies, cf. section 2.1. The VGG-16 topology has more than two-and-a-half times as many convolutional layers in comparison to the AlexNet one. The VGG-19 topology has three additional convolutional layers as the VGG-16 topology. The additional convolutional layers are added behind the existing ones by using the same kernel sizes to extend each convolution operation. After each additional convolutional layer of the VGG-16 and VGG-19 topology a normalisation layer is implemented in deviation from the original topology definitions. The expectation of the VGG-16 and VGG-19 is to obtain better results with a deeper CNN topology.

The four various input image sizes are: (64, 64, 3), (128, 128, 3), (256, 256, 3), and (512, 512, 3) RGB pixels. They vary logarithmically on the proposed input image size of 256 px by a factor of two. The expectation of the various input image sizes is that larger input images obtain better results due to feature details.

The results of this simulation study are listed in table 3.6. Accuracy was used as a metric for comparison with its 90 % confidence intervals. Each combination was simulated five times with random weight initialisation[25]. The mean accuracy results increase with a larger input image size until they stagnate after 256 px with 85.6 ±0.6 % to 84.2 ±2.5 % for 512 px. The VGG-16 topology achieves the most consistent results among all CNN topologies over all input image sizes with a mean accuracy of 85.2 %. The highest single results are achieved by the AlexNet topology together with an input image size of 256 px (86.4 ±0.5 %) and the VGG-16 topology with an input image size

[25]The weights are initialised between [0, 1] by using an uncorrelated equal distribution.

of 512 px (86.8 ±0.8 %) without a significant difference between both. Moreover, the 256 px input image size produces quite constant simulation results with the lowest 90 % confidence interval values. Disadvantages of the deeper CNN topologies and larger input image sizes are the longer training run-time and more expensive hardware resources. For example, processing of the VGG-16 CNN topology with an input image size of 512 px takes about five times longer than the proposed AlexNet topology with an input image size of 256 px. Moreover, twice as many hardware resources are required.

In conclusion, expanding the simulation study to other CNN topologies with higher complexity, i.e. more convolutional layers, does not achieve a significantl higher accuracy. Furthermore, this can only be achieved if larger input images and a longer training run-time with more hardware resources are applied in order to reach higher accuracy results in general. This relationship was also explained in subsection 2.1.4 concerning the hyper parameter setup and the three deep learning properties, cf. figure 2.5.

The AlexNet topology chosen in combination with the default (256, 256, 3) input image size yields the highest reliability, which is indicated by the lowest 90 % confidence interval with ±0.5 % accuracy. Hence, the proposed CNN topology and input image size used in the IEEE ITS paper have been retained to develop the deep metadata fusion approach in the IEEE RA-L paper, see chapter 4. This was carried out to ensure comparability between both manuscripts and under consideration of the plurality of simulations required to keep the training run-time within an adequate time window.

3.6.2 Investigation of Training Dataset Size

Using machine learning algorithms formerly raises the question of the overall quality of the dataset, i.e. the quantity, resolution, and diversity of the frames. This question will be answered with the following investigation of the training dataset size.

The training dataset ratio starts with 5 % and is doubled until 100 % is reached (100 % training dataset ratio relates all available 10,360 training dataset frames, cf. table 3.1 in subsection 3.4.1). Each training dataset ratio is simulated five times with random weight initialisation, again. A sample of all available frames is randomly selected[26] for each simulation. The test dataset is the same as in subsection 3.4.1 and is not changed during the simulations.

The simulation results are shown in figure 3.12, which uses the accuracy and precision metric for the evaluation. It exposes that a training dataset ratio < 20 % does not produce usable results. The accuracy value converges the minority event rate[27] of 53.3 %

[26]The training frames are selected by means of a random number generator between [1, 10360], which uses an uncorrelated equal distribution.

[27]The majority (relevant traffic lights) or minority (non-relevant traffic lights) event rate indicates the average distribution of relevant to non-relevant traffic lights in the database, cf. subsection 3.4.1.

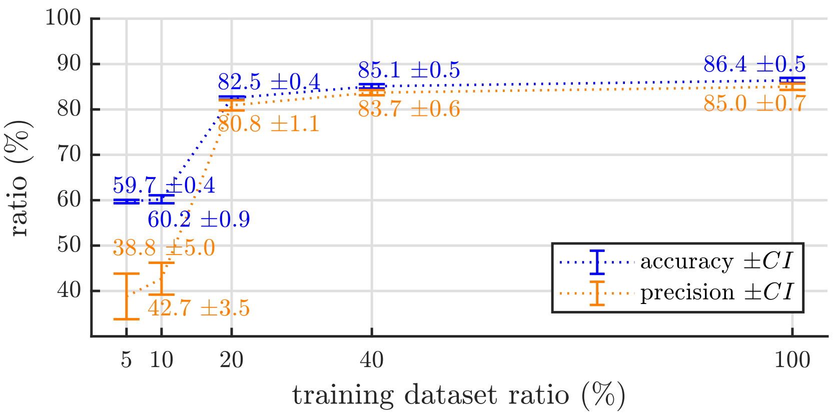


Fig. 3.12 Results of the training dataset size investigation. The training dataset is investigated by using different training dataset ratios between 5 % and 100 %. The results are displayed for the accuracy and precision metric by using the 90 % confidence interval CI. From a training dataset ratio of 40 %, the results (accuracy and precision metric) begin to saturate.

for non-relevant traffic lights and the precision value undercuts the event rate. Training dataset ratios $\geq$ 20 % produce usable results over 80 % accuracy and precision with decreasing differences between each doubling of the training dataset ratio. For example, the difference between the 40 % and 100 % training dataset ratios is only 1.3 % accuracy, which is still significant, see figure 3.12.

In summary, it exposes that a higher training dataset ratio produces a higher accuracy value than a smaller training dataset ratio. The results start to saturate from a training dataset ratio of approximately 40 % accuracy. It is assumed that increasing the training dataset would not produce significantly better results by extrapolating the accuracy and precision metrics curves over 100 % in figure 3.12. Consequently, the database contains sufficient images for the respective approach, which is based in total on 404[28] sequences recorded in four German cities with over 55 thousand associated relevant and non-relevant traffic lights.

3.6.3 Evaluation of the Traffic Light to Lane Association Main Challenges

Three main challenges regarding the traffic light to ego-vehicle lane association or assignment were mentioned in section 3.1. These three main challenges are:

1. All relevant traffic lights are rarely positioned above or beside their respective lanes.

[28]The complex dataset has 404 sequences with approximately 55 thousand relevant and non-relevant traffic lights and the full dataset has 848 sequences with approximately 105 thousand relevant and non-relevant traffic lights, see subsection 4.5.1.

2. Some traffic lights are positioned closely together but are not relevant for the same lane.
3. Traffic lights that are not relevant must be rejected, i.e. tram, pedestrian and bicycle lights as well as traffic lights that belong to the next upcoming or neighbouring intersection.

Figure 3.13 shows a histogram of all relevant and non-relevant traffic lights depending on their absolute lateral position centred on the ego-vehicle lane centre. In addition, the lane distances for the left and right neighbouring lanes are marked with a mean lane width of 3.5 m at the bottom of figure 3.13. This figure is used to explain the three main challenges more specifically:

- Not all relevant traffic lights are located directly in the lateral range of the ego-vehicle lane. On the contrary, they extend up to the fourth left and right neighbouring lanes (1. main challenge).
- For each lateral position of relevant traffic lights, there are also non-relevant traffic lights. This makes an assignment impossible based on the lateral traffic light positions (2. main challenge).
- The lateral exterior, other area, contains predominately non-relevant traffic lights, e.g. pedestrian or bicycle lights that must be rejected (3. main challenge).

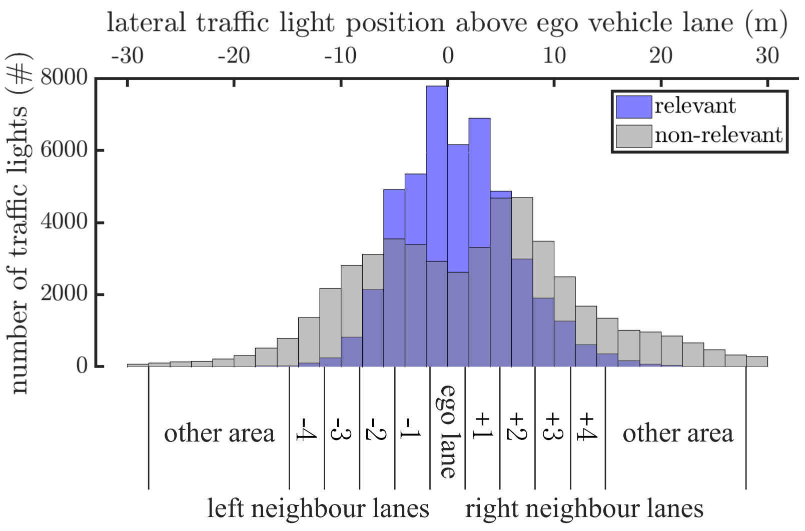


Fig. 3.13 Histogram of the lateral traffic light positions for relevant and non-relevant traffic lights with respect to the ego-vehicle lane line markings. Relevant as well as non-relevant traffic lights are located in the lateral range of the ego-vehicle lane. Non-relevant traffic lights occur exclusively in the other area lateral range.

Tab. 3.7 Evaluation of the three traffic light to lane association main challenges.

No.	Challenge Short Description	Average Accuracy
1.	Traffic lights are not always positioned over their corresponding lane.	86.3 %
2.	Some traffic lights are positioned close together but are not relevant for the same lanes.	71.1 %
3.	Non traffic lights, e.g. traffic lights for trams, pedestrians, bicycles, or for other intersections.	97.4 %

The three main challenges described are quantitatively evaluated to identify potential for improving the proposed approach. The corresponding results are displayed in table 3.7. The results are extracted from the test dataset by allocating each traffic light to be resolved to one of the main challenges. The table indicates the average accuracy of the proposed approach in the IEEE ITS paper for each main challenge.

The first main challenge achieves an accuracy of 86.3 %, which is in the average accuracy value range of the complete test dataset with 86.4 % $\pm$1.0 %. The second main challenge yields the lowest results with 71.1 % average accuracy and has the highest potential for further improvements. In contrast to this, the third main challenge yields the highest results with 97.4 % accuracy, see table 3.5.

3.7 Brief Discussion of the IEEE ITS Paper

In this section, a few specific aspects of the IEEE ITS paper are briefly discussed: the IPM transformation of the input image road part, the comparability between the approach and subjective test results, the evaluation of the results against the distance to the stop line, and the evaluation over the number of lanes with reference to the three traffic light to lane association main challenges.

The IPM transformation of the input image road part was investigated in subsection 3.4.3. It was discovered out that the approach reaches significantly higher results with this type of input image preparation, particularly for distances $\geq$ 50 m. This input image preparation supports the CNN training process as the functionality of CNNs is based on similar features. The IPM transformation of the road part leads to more symmetry between various input images. This produces more identical features, e.g. lane arrow markings and lane line markings, which also do not vary much in their size and orientation.

In subsection 3.4.4, a subjective test conducted was described that was used for a comparison to the respective approach. The subjective test uses 100 randomly selected images. These were taken from prepared CNN input images (down-sampled image resolution, increased and highlighted traffic lights, and IPM transformed road part). For this reason, conclusions can only be drawn in direct comparison to the respective approach. Statements about the general ability of humans to solve the TL2LA problem are not possible. This open question is followed up by another subjective test in the IEEE RA-L paper, see subsection 4.4.4, and is discussed in detail later on, see chapter 5.

The results of the approach were evaluated against the distance to the stop line and the number of lanes (in the same driving direction) of the intersection scenarios, cf. subsection 3.4.5. It was determined that accuracy increases with decreasing distance to the stop line because of the clearer lane courses and traffic light positions in the near field. The accuracy courses of the approach and subjective test coincide insignificantly with one exception for the CNN model at $25 \leq d < 50$ m, cf. figure 3.8. There, the approach outperforms the subjective test with $+4 \pm 1.2$ % accuracy. It is assumed that this exception is due to the test dataset distribution and the selected interval limits[29] for the evaluation.

The evaluation over the number of lanes exposed that intersection scenarios with five or more lanes achieve the lowest accuracy (78.7 %). This result relates to the first and second main challenges, which occur more frequently at large intersections. In other intersection scenarios with two, three, or four lanes, the third main challenge also occurs, cf. table 3.7. This main challenge increases the results for these three lane groups to approximately 88 % accuracy, cf. figure 3.9. At this point, it can be anticipated that the following deep metadata fusion approach, cf. chapter 4, resolves all three main challenges with minor differences[30]: 90.1 % (1. main challenge, 76.3 % (2. main challenge), and 97.9 % (3. main challenge) average accuracy. However, the second main challenge is still the most demanding one and is sometimes also a challenge for humans.

[29]The accuracy course of the only-vision approach and the complex test dataset (same approach and dataset) has no exceptions with 15 m interval limits, see figure 4.9 (b) in subsection 4.5.5.

[30]The results correspond to the only-vision approach results listed in table 4.4 for the complex test dataset.

4 Manuscript: Deep Metadata Fusion for Traffic Light to Lane Assignment, IEEE Robotics and Automation Letters (IEEE RA-L)

Exposition The following chapter deals with my "Deep Metadata Fusion for Traffic Light to Lane Assignment" paper. It was published in the IEEE Robotics and Automation Letters in 2019. This IEEE RA-L paper presents an innovative deep metadata fusion approach that is applied in the research field of computer vision and describes a method to fuse heterogeneous metadata with image data. It follows up on the approach of the preceding IEEE ITS paper in chapter 3 and pursues the same goal: traffic light to lane association or assignment, respectively. Again, the focus is on the traffic light to ego-vehicle lane assignment, but the approach proposed is also applied to the left and right neighbouring lanes. The results are compared to several other approaches as well as against a redesigned subjective test with more subjects, test sequences, and a realistic visualisation.

The already presented IEEE ITS CNN model approach of the previously discussed manuscript, in chapter 3, is referred as only-vision approach in this chapter as well as for the remainder of this thesis. This only-vision approach forms the basis for the deep metadata fusion approach presented. The deep metadata fusion approach employs three out of four working steps of the only-vision approach: input image preparation[31], output vector mapping, and simple majority over frame decision maker. The last two listed working steps form the integral post-processing of the proposed deep metadata fusion approach. The deep metadata fusion approach also uses the same CNN output, which is renamed from output regression vector to output indication vector in this chapter as well as for the remainder of this thesis. In addition, the presented dataset with exclusive complex intersection scenarios is reused and is referred to as a complex test dataset.

[31]The input image preparation takes over the IPM road part transformation, but it does not take over the increased and highlighted traffic light positions. This is not required any more due to the use of traffic lights as metadata.

Likewise, the distribution of the training and test datasets remained as it to ensure the comparability between the results of both manuscripts. The dataset is extended with less complex intersection scenarios (called full dataset) and is enriched with the annotated metadata (lane arrow markings, IPM lane line markings, explained in subsection 2.2.5, and lane signs) introduced in chapter 1. Moreover, the deep metadata fusion approach applies a neural network merge layer for the fusion operator, cf. subsection 2.1.3, and uses the explained data normalisation types, cf. subsection 2.1.5 to enable a deep fusion with the annotated metadata.

The following contents of this chapter 4 were taken identically from the original publication, see [7], and were adapted to the format of my dissertation.

Abstract We present a deep metadata fusion approach that connects image data and heterogeneous metadata inside a Convolutional Neural Network (CNN). This approach enables us to assign all relevant traffic lights to their associated lanes.

To achieve this, a common CNN topology is trained by down-sampled and transformed input images to predict an indication vector. The indication vector contains the column positions of all the relevant traffic lights that are associated with lanes. In parallel, we fuse prepared and adaptively weighted Metadata Feature Maps (MFM) with the convolutional feature map input of a selected convolutional layer.

The results are compared to rule-based, only-metadata, and only-vision approaches. In addition, human performance of the traffic light to ego-vehicle lane assignment has been measured by a subjective test.

The proposed approach outperforms all other approaches. It achieves about 93.0 % average precision for a real world dataset. In a more complex dataset, 87.1 % average precision is achieved. In particular, the new approach reaches significantly higher results with 93.7 % to 91.0 % average accuracy for a real world dataset in contrast to lower human performance.

Keywords Intelligent Transportation Systems, Computer Vision for Transportation, Deep Learning in Robotics and Automation

4.1 Introduction

Traffic Light to Lane Assignment (TL2LA) is a necessary function for Traffic Light Assistance Systems (TLA) as well as for autonomous driving in urban environments. Some upcoming applications are braking at stop lines or visualising the current traffic light colour state to prevent drivers from crossing an intersection. All these applications

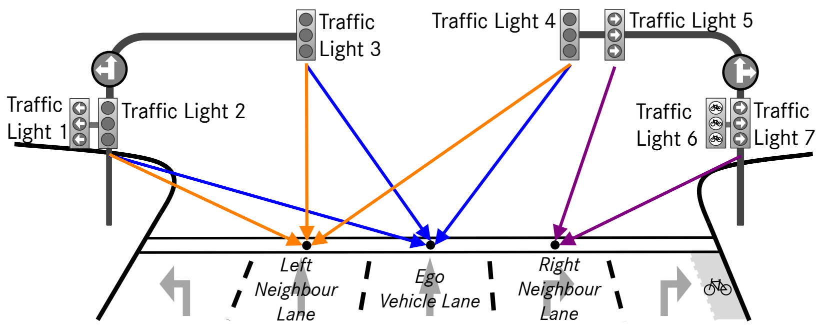


Fig. 4.1 Generic example of the traffic light to lane assignment function for the ego-vehicle lane (blue arrows), left neighbour lane (orange arrows), and right neighbour lane (purple arrows).

have in common that reliable information about the connection between a traffic light and its lane is required.

The TL2LA function will help to avoid accidents and save lives in future. Today, about 4.0 % of all (≈2.600) fatalities per year are caused by motorcar accidents with a red light offence in Germany [8], [9].

Our paper scope is to assign traffic lights to the ego-vehicle lane as well as to the left and right neighbour lanes. This is illustrated in figure 4.1, which visualises the TL2LA function. Moreover, this figure also underlines the challenge that relevant traffic lights are not always mounted directly above their associated lane, see traffic light 4.

This paper introduces a Convolutional Neural Network (CNN) with deep metadata fusion for producing traffic light to lane assignment decisions based on single frames. In the present work metadata are produced by use of a camera sensor. Alternatively, metadata could be generated also by other sensors like RaDAR or LiDAR [74], [75]. One additional important aspect is that our approach works independently of any prior knowledge about the intersection (e.g. cloud map data or statistical observations).

The paper is structured into five main sections: related work (sec. 4.2), our proposed approach (sec. 4.3), other approaches for comparisons (sec. 4.4), experiments and evaluation (sec. 4.5), and a conclusion (sec. 4.6).

4.2 Related Work

The problem of TL2LA belongs to the large field of autonomous driving, which has become a very important aspect in autonomous robotics. As such it is highly interdisciplinary. Hence, the related work is grouped into three parts. 1) Traffic Light Recognition (TLR) and Lane Detection (LD). These two research fields deliver the essential metadata for our deep fusion approach. 2) Approaches to develop a TL2LA function. And 3)

Fusion in CNNs, which relates to the specific method that we have chosen to address the TL2LA problem.

4.2.1 Traffic Light Recognition (TLR) and Lane Detection (LD)

In general, TLR can be divided into two categories of approaches. The first category relates to feature based detectors. In [16] and [17] a support vector machine takes over the main part of the detection algorithm. The approach in [20] introduces methods to detect traffic lights by use of stereo image data. Another approach [21] compares different feature detectors like spotlight, colour or circle detectors. The second category contains pure deep learning approaches [15], [22], [23]. They all use CNN based approaches with different specific network architectures to detect and classify traffic lights.

The second necessary building block concerns lane detection (LD). Lane line and lane arrow markings deliver important information about the intersection layout and are often constrained to urban environments, [24]–[26]. Other universal approaches are also possible, i.e. to detect and classify different kinds of lane markings like crossings, stop lines, and arrows [27].

4.2.2 Traffic Light to Lane Assignment (TL2LA)

In the context of Advanced Driving Assistance Systems (ADAS) this topic was specifically pointed out as an unsolved problem, [70]. A simple procedure uses stored traffic light relevance information in a database as a look up table for vector maps [71]. An approach using a rule-based routine assumes that the largest and nearest traffic light in driving direction is relevant and they improve their hypothesis statistically by driving over the same intersections many times in [72] (Traffic Light Mapping). Another rule-based approach is presented in [73] (Main Traffic Light). A relevant traffic light is selected using the following condition: it has to be the traffic light with the largest size and highest position from the group with most traffic lights of the same colour.

The newest approach only uses vision and associates relevant traffic lights to the ego-vehicle lane. They develop a CNN model that is fed with down-sampled camera input images and trained on a target regression vector. Our new deep metadata fusion approach starts from this idea, which we had also pursued in an early study [66].

4.2.3 Fusion in Convolutional Neural Networks

Fusion approaches with CNNs can be divided into Multiscale Feature Fusion (MFF) and Deep Fusion (DF) approaches.

The MFF technique is based on spreads or the duplication of convolutional features maps, respectively. Then, these feature maps are differently processed with additional layers. Afterwards, feature maps are merged again, which represents the actual fusion. MFF-CNN approaches have been used for different approaches like face recognition, [55], [76] and pedestrian detection, [54] and showed high performance. Moreover, GoogLeNet [31] and ResNet [32] can be defined as MFF-CNNs, due to their inception module or building block architecture, respectively.

The term Deep Fusion (DF) describes techniques to combine heterogeneous data sources by use of a CNN. DF is applied in the field of computer vision, e.g. image data and point cloud depth maps are fused for a semantic segmentation [77], salient object detection [78], or semantic event recognition of sequences [79]. Semantic event recognition in [79] is achieved by fusing three CNN pathways (action, object and scene) to interpret and classify the sequence event. In the field of ADAS, [60] introduces three approaches to fuse heterogeneous data from a RaDAR, LiDAR and camera sensor to increase object detection performance.

4.3 Proposed Approach

4.3.1 Overview

Our main contribution is a novel DF approach to combine input image data with additional heterogeneous metadata in a CNN. The approach creates adaptively weighted Metadata Feature Maps (MFM) from the heterogeneous metadata. These MFMs are fused with the maximum pooled convolutional feature map output of a previous CNN layer. Fusion is executed by an element-wise multiplication between the MFMs and the convolutional feature maps. Figure 4.2 exemplifies schematically the deep metadata fusion approach for a selected convolutional fusion layer (Conv. 2).

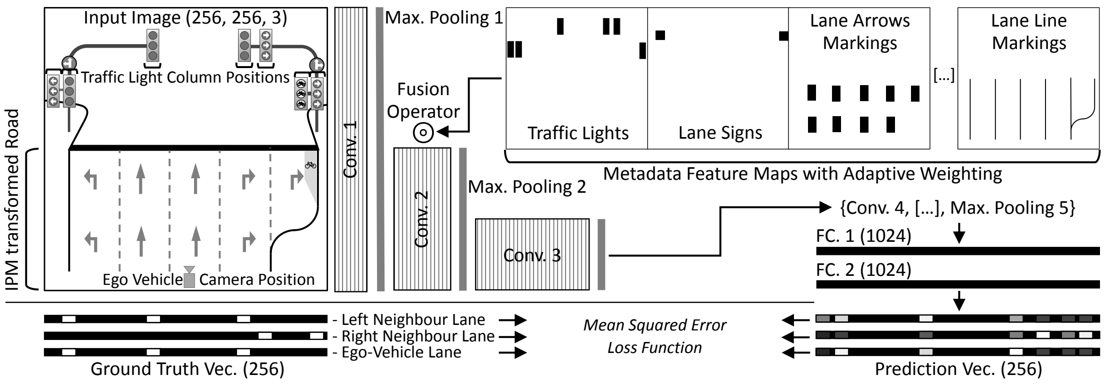


Fig. 4.2 Schematic illustration of our deep metadata fusion approach for convolutional layer 2 (Conv. 2) with Metadata Feature Maps (MFM), input image, and indication vectors.

4.3.2 Convolutional Neural Network Input and Output

Figure 4.2 shows a generic input image, which corresponds to the TL2LA example from fig. 4.1. All input images are down-sampled and cut to size 256 x 256 in an RGB colour space. The road in the bottom half of an input image is transformed into an Inverse Perspective Mapping (IPM) image using camera parameters and a flat world assumption, cf. [61]. This IPM image contains up to a maximum of the next 50 meters of the road from the ego-vehicle location to the stop line of the intersection. Thus, the input image becomes insensitive to asymmetric perspective distortions of the lane line and the lane arrow markings, cf. [66].

We train our CNN to predict an indication vector (Prediction Vec., see fig. 4.2.), the length of which is equal to the input image width. The target indication vector (Ground Truth Vec., see fig 4.2.) contains the column positions of all relevant traffic lights for an associated lane. If a traffic light is relevant for its associated lane, the target indication vector will be set to one at this position. We create and train our CNN on three of these target indication vectors: one for the ego-vehicle lane and one for each neighbour lane.

4.3.3 Metadata Feature Map Preparation

The annotated metadata are available in the following formats: annotated rectangles in recorded frames for traffic lights, lane signs, and lane arrow markings and annotated polylines in IPM transformed frames for lane line markings. During online operation these MFMs are produced by existing perception systems, e.g. LD or TLR functions in sec. 4.2.1. Using this, we prepare our MFMs in a compatible data format with the CNN to enable fusion between the convolution layer outputs and heterogeneous metadata.

In total we prepare twelve MFMs. Figure 4.3 visualises the corresponding MFMs for an example input image. An MFM frame is a binary-valued image of the same size as the input images. Thus, all MFMs values correspond to the image input pixels and their

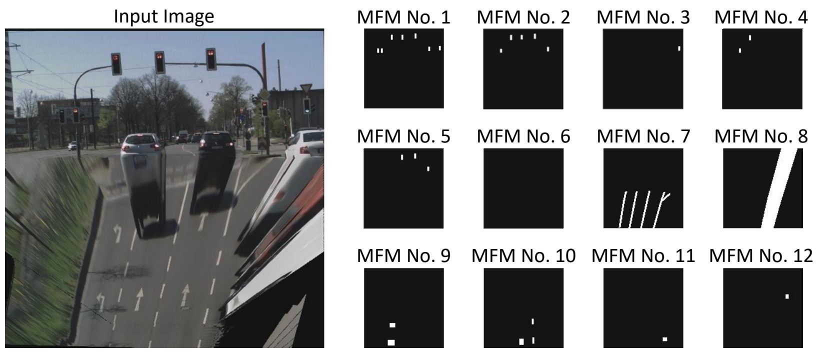


Fig. 4.3 Example of an input image with its twelve corresponding metadata feature maps (MFM). The MFM No. and descriptions are listed in table 4.1.

Tab. 4.1 Metadata feature map (MFM) list of the deep metadata fusion approach.

MFM No.	Description
1	All traffic lights
2	Green or yellow traffic lights
3	Red or red-yellow traffic lights
4	Traffic lights with left direction arrow
5	Traffic lights with straight direction arrow
6	Traffic lights with right direction arrow
7	All available lane line markings
8	Ego-vehicle, left or right neighbour lane line markings
9	Lane arrow marking in left direction
10	Lane arrow marking in straight direction
11	Lane arrow marking in right direction
12	All lane signs

locations.

By default all values are set to zero. The MFMs for traffic light and lane sign features (MFM No. 1, 2, 3, 4, 5, 6, and 12 in table 4.1.) are generated by setting their annotated rectangle positions to one. The MFMs for lane marking arrows (MFM No. 9, 10, and 11 in table 4.1.) are generated identically with the difference that the annotated rectangle positions are converted according to the IPM. The lane line marking features (MFM No. 7 and 8 in table 4.1.) are generated by marking all polylines with one in the MFM with a fixed line width of twelve pixels. In addition, the area between the left and right ego-vehicle lane line markings of MFM No. 8 is set to one and the area is extrapolated until the top of the MFM. Note, MFM No. 8 changes depending on the associated ego-vehicle, left or right neighbour lane.

The MFMs are down-sampled to the output size of the previous maximum pooling layer by nearest neighbour interpolation before fusion with the selected convolutional layer. We define a metadata feature map tensor $A^{\mathrm{l,e}}_{\mathrm{p,q}}$, which contains all MFMs of the selected convolutional fusion layer l. The feature map spatial position is given by p and q while the MFM number (of all $E = 12$ MFMs) is indicated by e[32].

[32]The MFM number E and MFM index e are said to be F and f in the originally published IEEE RA-L paper. The variable names were changed to avoid confusion with previously defined variable names in this dissertation.

4.3.4 Adaptive Weighting of Metadata Feature Maps

The activations in the convolutional layer and the metadata feature maps do not necessarily lie in a similar value range. To prevent one from outweighing the other, values need to be scaled, which we explain in the following.

In the CNN, an output tensor $Z^{\mathrm{l,k}}$ at layer l for feature map k is obtained by filtering the input tensor X^{l} with a kernel $W^{\mathrm{l,k}}$, adding a bias term b^{l} and then applying an activation function. In the following, we will refer to $Z^{\mathrm{l,k}}_{\mathrm{p,q}}$ in order to denote the activation in layer l, feature map k at position (p, q), cf. [38] for details.

The adaptive weighting of the MFMs is executed by multiplying the metadata tensor $A^{\mathrm{l,e}}_{\mathrm{p,q}}$ with the global average g^{l}. The adaptive weighting technique is similar to layer normalisation (to create g^{l}) and instance normalisation (to adapt the MFMs), cf. [50]. The global average $g^{\mathrm{l}} = \bar{Z}^{\mathrm{l}}$ is calculated over all activations (p, q) of all convolutional feature maps k. Hence, g^{l} represents the average activation within the CNN layer after a ReLU activation. This is done for each forward propagation step during training and testing of the CNN. We obtain the deep metadata fusion tensor

$$M^{\mathrm{l,k}}_{\mathrm{p,q}} = \begin{cases} A^{\mathrm{l,e}}_{\mathrm{p,q}} \cdot g^{\mathrm{l}} & \text{for } \mathrm{k} \in \{1, ..., E\} \text{ with } \mathrm{e} = \mathrm{k} \\ 1 & \text{for } \mathrm{k} \in \{E+1, ..., K\}. \end{cases} \tag{4.1}$$

This tensor has to have the same size of dimensions as the output tensor $Z^{\mathrm{l,k}}_{\mathrm{p,q}}$ of the selected convolutional fusion layer $(\mathrm{l}+1)$. The case differentiation regarding k is made, because the total number of convolutional feature maps K is not equal to the number of metadata feature maps E depending on the selected convolutional fusion layer. Hence, we fill the remain of $M^{\mathrm{l,k}}_{\mathrm{p,q}}$ with ones. We make sure that the MFMs are at the same index position k of $M^{\mathrm{l,k}}_{\mathrm{p,q}}$ for each forward propagation. In consequence, we always trim the same convolutional feature map on the same fused MFM during the back propagation.

4.3.5 Deep Metadata Fusion in the CNN

Deep metadata fusion is achieved by means of a multiplication layer, which is able to merge different tensors of the same size. We use the element-wise product $\circ$ as fusion operator, which results from the multiplication of the tensors $Z^{\mathrm{l,k}}_{\mathrm{p,q}}$ and $M^{\mathrm{l,k}}_{\mathrm{p,q}}$. As result we get the input

$$X^{\mathrm{l+1,k}}_{\mathrm{p,q}} = Z^{\mathrm{l,k}}_{\mathrm{p,q}} \circ M^{\mathrm{l,k}}_{\mathrm{p,q}} \tag{4.2}$$

for the next convolutional layer. This is motivated for the CNN in order to let it learn from the metadata encoded in the MFMs together with the $(K - E)$ not fused convolutional feature maps.

The CNN used for the deep metadata fusion approach is schematically illustrated in figure 4.2. It consists of five successive blocks. Each block possesses a convolutional layer (Conv.) with 3 by 3 shared weight kernel filters, a normalisation layer and a maximum pooling layer (Max. Pooling) with a symmetric down-sample factor of two. Thus, the first block has an output size of (128, 128, 32) and the last block has an output size of (8, 8, 512). These blocks are followed by two fully connected layers (FC.), which are preceded by dropout layers. The CNN is initialised with random weights.

We interpret the problem as a regression problem and consequently employ the mean squared error as loss function. The CNN is trained with the stochastic gradient descent momentum algorithm (SGDM). We train the CNN using early stopping with a maximum of 100 training epochs. If the validation error stagnates or increases during the training for more than five epochs, training will be terminated.

4.3.6 Post-Processing

We implement two post-processing steps that improve temporal consistency. First, we extract single traffic light results from each output indication vector for evaluation. Second, we apply a simple majority over frame decision maker on all single traffic light results. This reduces the number of flipping single traffic light results for the same traffic light in consecutive frames or output indication vectors respectively, cf. [66].

4.4 Approaches for Comparisons

4.4.1 Rule-Based Approaches

A straightforward way of determining relevant traffic lights of the ego-vehicle is to rely on a set of heuristics. The advantage of these approaches is that they do not need a training dataset, they are simple to implement, and fast in execution.

Two known approaches are the so called Traffic Light Mapping approach, cf. [72], and the Main Traffic Light approach, cf [73]. These two approaches and their rules have already been described in sec. 4.2.2.

A third one is the Traffic Light above Ego-Vehicle Lane approach. This is another rule-based approach. Its rule is as follows: A laterally mounted traffic light between the left and right ego-vehicle lane line markings is relevant. If no traffic light is within the ego-vehicle lane line markings, the traffic light with the shortest lateral distance to the ego-vehicle lane centre will be selected as relevant.

4.4.2 Only-Metadata Approach

We implemented a machine learning approach to assess the question whether a camera image is actually necessary for the task. The approach uses a Multi-Layer Perceptron

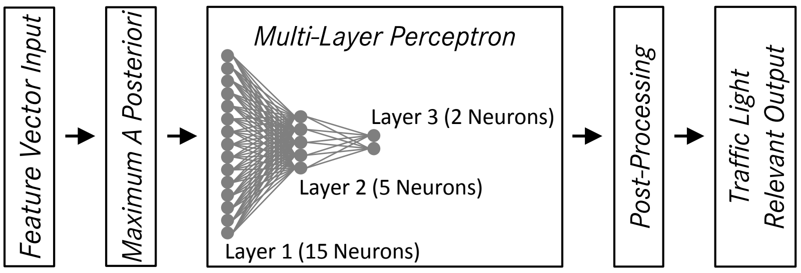


Fig. 4.4 Overview of the only-metadata machine learning approach.

(MLP) to process our annotated metadata. The main difference to our proposed approach is that we do not use any image data or fusion technique to assign relevant traffic lights to their lanes.

Figure 4.4 explains our only-metadata approach. A feature vector containing 31 features is used as input data, see table 4.2. Each feature vector represents one traffic light of the current frame, which is classified as being relevant or non-relevant. The temporal consistency of some of the features (No. 1, 2 3, 7, 8, 9, 20, 21, 22, 26, 27, 28 in table 4.2) is improved by a Maximum A Posteriori (MAP) estimation. This MAP estimation takes all features from the previous frames of the sequence into account to correct outliers. These outliers can occur due to the disparity calculation, which is used to estimate object positions of the metadata. The feature vector is processed by a

Tab. 4.2 Feature list of the only-metadata approach.

No.	Description
1-3	lateral, longitudinal, and vertical traffic light position
4	traffic light colour (unknown, red, yellow, green, red-yellow) coding
5-6	global traffic light height and width
7-9	mean lateral, longitudinal, and vertical traffic light assembly position
10-12	traffic light left, straight, and right direction arrow type coding
13-15	current lane left, straight, and right direction type coding
16-17	number of left and right neighbour lanes
18-19	left-sided and right-sided lateral ego-vehicle lane line marking position below traffic light assembly
20-22	nearest lateral, longitudinal, and vertical lane sign position
23-25	nearest lane sign left, straight, and right direction type coding
26-28	nearest lateral, longitudinal, and vertical lane arrow position
29-31	nearest lane arrow left, straight, and right direction type coding

trained MLP, which consists of three layers. The output layer uses a softmax activation function. Both other layers use a sigmoid activation functions.

The MLP is trained with the SGDM back propagation algorithm and uses cross entropy as loss function to classify the input feature vectors. We train the MLP for a maximum of 100 epochs with early stopping, see sec. 4.3.5. The output of the MLP is post-processed with the same simple majority over frame decision maker as for our deep metadata fusion approach, see sec. 4.7.1.

4.4.3 Only-Vision Approach

The opposite problem to the metadata approach is to rely exclusively on vision. To consider this setting, we implemented a vision based approach, as described in section 4.2.2. This approach uses only the input image and no further metadata. It makes use of the same CNN topology, parameter setup, input images, and target indication vectors as our proposed approach. One difference is that all traffic lights in the input images are increased threefold in size and highlighted with a rectangle box of the corresponding traffic light colour. This preprocessing is intended to help the CNN adapting to the traffic light position by giving steeper gradients, cf. [66].

4.4.4 Human Performance

In order to yield a valid baseline to evaluate the algorithm's performance, we conducted a subjective test involving human raters. The subjective test is designed to measure human performance at assigning all relevant traffic lights to the ego-vehicle lane. The tool for the subjective test is shown in figure 4.5.

We randomly selected from each of our 138 test dataset sequences two different final video frames (the database is explained in sec. 4.5.1). In sum, we showed our subject group 276 intersection scenarios with a maximum video frame resolution of (1792, 896, 3) RGB pixels.

The subject group were presented a short video with a length of two seconds of each intersection scenario, which represents a covered driving route of about 25 meters. Following this, the subject group had ten seconds time to assess all relevant traffic lights for the ego-vehicle lane in the last frame of the video. All previously annotated traffic lights are marked with a rectangle box with the same colour as the respective traffic light, i.e. the subjects did not have to recognise the traffic lights themselves. A driver would normally have about 7 seconds time for 100 meters driving route to a traffic light assembly at the end of an intersection. So we allowed here more time to get an upper bound for the human performance.

The subject group consists of 32 men and 8 women with an average age of 32.3 years

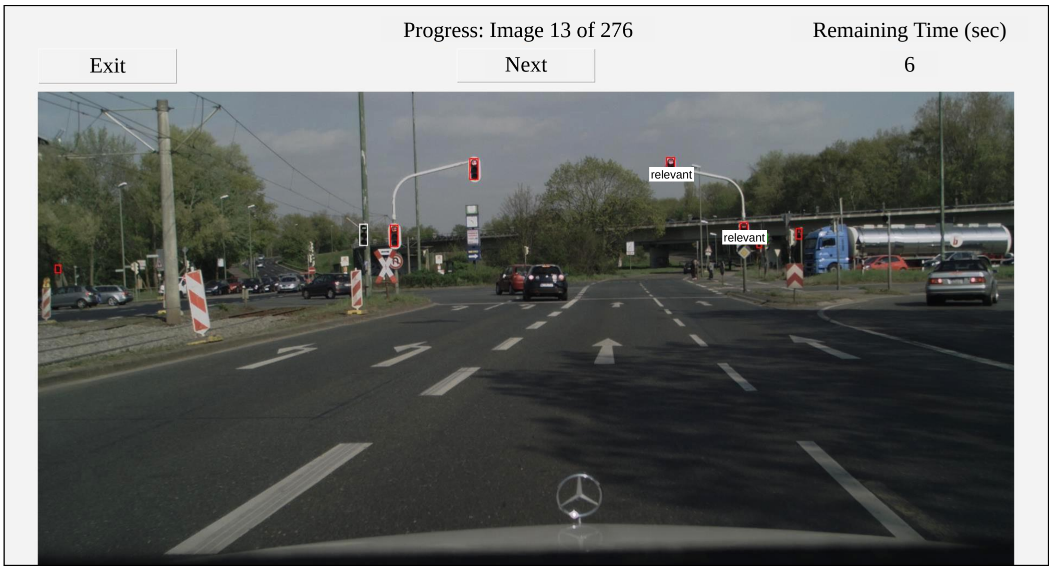


Fig. 4.5 Screenshot of our programmed subjective test tool (engl. version). The subject group had the task to mark all relevant traffic lights for the ego-vehicle lane with a relevant tag by clicking on it.

and 9.8 years average experience in road traffic. Each test could be paused and continued later to avoid tiredness of the subjects. The maximum test duration was 40 minutes. The mean assessment time of a video was found to be 5.23 seconds. This shows that the constraint of the assessment time to ten seconds time is negligible.

4.5 Experiments and Evaluations

In the evaluations we focus on the traffic light to ego-vehicle lane assignment, because this lane is the most important one for TLA applications and for reasons of clarity and comprehensibility. We use as evaluation metrics

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}\ , \tag{4.3}$$

$$precision = \frac{TP}{TP + FP} \text{ and} \tag{4.4}$$

$$F_1 score = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{4.5}$$

They are defined by use of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) assigned relevant and non-relevant traffic lights.

For all experiments the same database was used, which is described in the next sec. 4.5.1.

4.5.1 Database

Images with a resolution of two megapixel and annotated traffic light metadata are used for our experiments from the DriveU dataset [12]. Furthermore, we have annotated the left camera images with additional metadata: lane arrow markings, lane signs, and lane line markings for all visible lanes, cf. fig. 4.6. The lane line marking metadata were annotated by use of an IPM panorama of each sequence, cf. fig. 4.6 (b), which was processed by frame stitching of the IPM-transformed frames. Moreover, we have made a complete traffic light to lane assignment for every lane of the considered intersection, cf. coloured assignment lines in fig. 4.6 (a). These lines represent the annotated ground truth for the traffic light to lane assignments.

We have 848 annotated real world sequences (full) with at least two to six lanes for the own direction of travel. 404 of these sequences represent complex intersection scenarios (complex), see table 4.1. Complex sequences are characterised by multiple possible driving directions and challenging traffic light arrangements. The full sequences represent a normal distribution of available intersection scenarios with the restriction that all intersections with only one lane are sorted out. Each sequence shows a drive through an intersection scenario until the vehicle reaches the stop line. All sequences are recorded in German cities. We split our database in two datasets, the full and the complex one, which are both divided into a training and test dataset. Note, the full and complex test datasets are disjoint and 10 % of the training data are used as validation data. The number of all annotated dataset frames and the number of all annotated metadata in these frames are displayed in table 4.3. The ratio of relevant to non-relevant traffic lights associated to the ego-vehicle lane is about 52 % to 48 % in the test datasets.

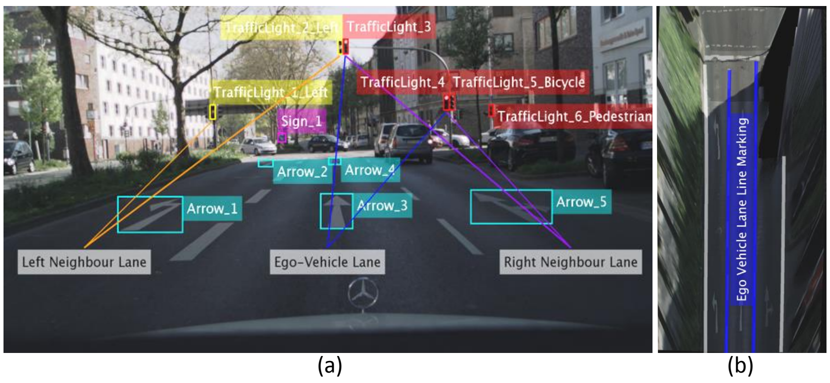


Fig. 4.6 One annotated frame with traffic lights, lane arrow markings, lane signs in (a), and lane line markings in the IPM-transformed frame in (b). The blue, orange, and purple lines in (a) represent the annotated ground truth for the traffic light to ego-vehicle lane, left neighbour lane, and right neighbour lane assignments.

Tab. 4.3 Number of elements in our complex (complex) and real world (full) training and test datasets.

	Training complex / full	**Test complex / full**
Sequences	356 / 758	48 / 90
Frames	32041 / 42754	1336 / 2563
Traffic Lights	49379 / 93110	7136 / 11574
Lane Arrow Markings	26138 / 38229	3311 / 5235
Lane Line Markings	62090 / 113786	8400 / 14490
Lane Signs	8857 / 15609	1373 / 1957

4.5.2 Investigations of Fusion Approach

The metadata fusion layer can be integrated into the CNN at each convolutional layer. In order to find out where the fusion of metadata is most effective, we conduct an experiment. We take the full dataset and simulate our approach for each possible deep metadata fusion layer of our common CNN topology ten times. The mean accuracy results are shown in figure 4.7. It turns out that the deep metadata fusion is more successful in the Conv. 1 to Conv. 3 layers. The accuracy decreases for deeper layers, see Conv. 4 and Conv. 5. We assume that is due to their smaller kernel feature map sizes. We recommend to make deep metadata fusion in Conv. 1 to Conv. 3. The input (fig. 4.7) represents a vanilla concatenation of the image with the MFMs, because a fusion between twelve MFMs and three RGB input feature maps is not possible. *Vanilla concatenation* means that we simply attached to the existing K feature maps or RGB input feature maps the $E = 12$ adaptively weighted MFMs, respectively.

Moreover, we investigated the used fusion operator. We exchanged the element-wise product against an element-wise sum and the vanilla concatenation operation. The results of these experiments are shown in table 4.4. Accuracy, precision, and F_1score

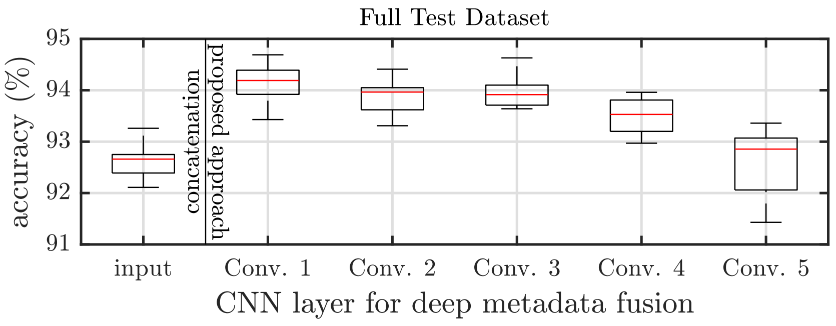


Fig. 4.7 Deep metadata fusion simulations with different fusion layers.

show only small differences (-0.4 % to -0.1 %) for the full dataset, which are not significant. However the results significantly decrease by up to 2.9 % precision with vanilla concatenation for the complex dataset. In general, all metrics decrease around 2 % by the use of the two other fusion operators for the complex dataset.

The mean test time of a single frame with relevant and non-relevant traffic lights was between ten and two milliseconds depending on the selected fusion layer with a NVIDIA GeForce GTX 1080 Ti.

4.5.3 Investigation of Metadata Feature Map Effect

We carried out an ablation study experiment to determinate the effect of each of the twelve fused MFMs. Twelve cases, each with the omission of one out of all possible twelve MFMs, are simulated. Figure 4.8 displays the negative impact on accuracy for each missing metadata feature map tested for the complex dataset. MFM No. 1 (all traffic lights) and MFM No. 8 (ego-vehicle lane line markings) have the highest impact on the achieved results. MFM No. 6 (traffic lights with right direction arrow) shows no significant impact. This may be because the information content of the MFM No. 6 is almost redundant relative to MFM. No. 4 and 5. Nevertheless, we recommend to use always all available MFMs to gain the best results because the omission of even one MFM has always a negative effect, cf. fig. 4.8.

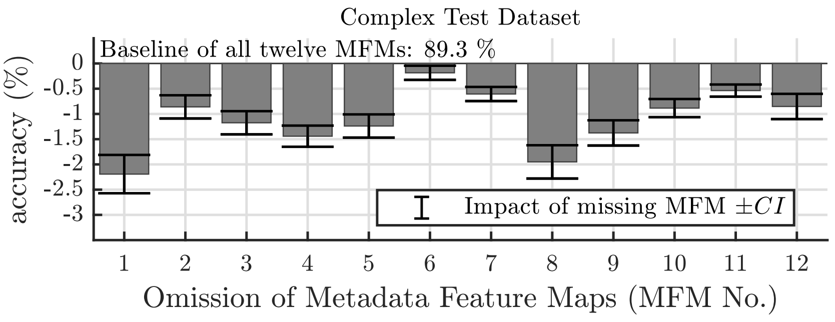


Fig. 4.8 Ablation study experiment to determinate the impact of each MFM.

4.5.4 Comparisons of different Approaches

Our proposed deep metadata fusion approach in sec. 4.3 and all other approaches introduced in sec. 4.4 were compared against each other.

We calculated mean accuracies, precisions, and F_1scores based on all single relevant traffic light results for both test datasets. In addition, we determined the 90 % confidence interval CI to make statements about significance. The subjective test uses the subject group with 40 participants as basis for the confidence interval. The machine learning,

Tab. 4.4 Mean results of our proposed approach and of the other approaches for the traffic light to ego-vehicle lane assignment.

Approach Name	**Accuracy** $\pm CI$ **complex / full**
Traffic Light Mapping [72][1]	-
Main Traffic Light [73][1]	-
Traffic Light above Ego-Vehicle Lane[1]	-
Only-Metadata Approach[2]	86.3 ±0.7 % / 86.9 ±0.4 %
Only-Vision Approach [66][3]	86.4 ±0.2 % / 91.6 ±0.2 %
Human Performance[4]	93.9 ±0.7 % / 91.0 ±0.9 %
Deep Metadata Fusion[5] (proposed approach)	89.3 ±0.4 % / 93.7 ±0.3 %
Deep Metadata Fusion[5] (element-wise sum)	87.6 ±0.7 % / 93.3 ±0.3 %
Deep Metadata Fusion[5] (vanilla concatenation)	87.3 ±1.0 % / 93.6 ±0.5 %
Deep Metadata Fusion[5] - Left Neighbour Lane	81.2 ±0.8 % / 82.9 ±0.8 %
Deep Metadata Fusion[5] - Right Neighbour Lane	83.1 ±0.3 % / 93.6 ±0.2 %
Approach Name	**Precision** $\pm CI$ **complex / full**
Traffic Light Mapping [72][1]	62.4 % / 77.9 %
Main Traffic Light [73][1]	56.2 % / 79.0 %
Traffic Light above Ego-Vehicle Lane[1]	57.1 % / 60.5 %
Only-Metadata Approach[2]	77.2 ±1.3 % / 88.8 ±0.6 %
Only-Vision Approach [66][3]	82.7 ±0.3 % / 91.9 ±0.3 %
Human Performance[4]	89.2 ±1.5 % / 92.9 ±1.3 %
Deep Metadata Fusion[5] (proposed approach)	87.1 ±0.6 % / 93.0 ±0.2 %
Deep Metadata Fusion[5] (element-wise sum)	84.5 ±0.6 % / 92.7 ±0.5 %
Deep Metadata Fusion[5] (vanilla concatenation)	84.2 ±0.9 % / 92.9 ±0.3 %
Deep Metadata Fusion[5] - Left Neighbour Lane	75.9 ±0.5 % / 78.4 ±0.8 %
Deep Metadata Fusion[5] - Right Neighbour Lane	81.0 ±0.6 % / 90.7 ±0.7 %
Approach Name	$\mathbf{F_1}$**score** $\pm CI$ **complex / full**
Traffic Light Mapping [72][1]	-
Main Traffic Light [73][1]	-
Traffic Light above Ego-Vehicle Lane[1]	-
Only-Metadata Approach[2]	82.0 ±0.9 % / 86.6 ±0.6 %
Only-Vision Approach [66][3]	82.1 ±0.4 % / 93.2 ±0.2 %
Human Performance[4]	90.9 ±1.2 % / 89.9 ±1.3 %
Deep Metadata Fusion[5] (proposed approach)	86.2 ±0.8 % / 94.0 ±0.1 %
Deep Metadata Fusion[5] (element-wise sum)	84.4 ±0.9 % / 93.6 ±0.4 %
Deep Metadata Fusion[5] (vanilla concatenation)	84.4 ±0.9 % / 93.9 ±0.4 %
Deep Metadata Fusion[5] - Left Neighbour Lane	76.6 ±0.4 % / 83.1 ±0.5 %
Deep Metadata Fusion[5] - Right Neighbour Lane	80.3 ±0.4 % / 93.9 ±0.3 %

[1]rule-based, [2]machine learning, [3]deep learning, [4]subjective test, [5]deep fusion

deep learning and deep fusion approaches employ ten trained and tested models with random initialisation and validation data as confidence interval basis. In contrast to this, the rule-based approaches have no confidence intervals because they are based on analytic functions. Moreover, the rule-based approaches are only evaluated with the precision metric, because they do not deliver a relevant or non-relevant traffic light decision for all traffic lights in a test frame.

Summarised results are given in table 4.4. Human performance achieves the highest mean accuracy, which is significant with 93.9 ± 0.7 % and a mean precision with 89.2 ± 1.5 % for the complex test dataset. Our new deep metadata fusion approach outperforms human performance with 93.7 % to 91.0 % mean accuracy as well as 94.0 % to 89.9 % mean F_1score and achieves equally good results with 93.0 ± 0.2 % to 92.9 ± 1.3 % mean precision for the full test dataset. The only-vision and only-metadata approaches obtain significantly lower accuracy and precision results than our proposed approach. In general, the only-vision approach performs significantly better than the only-metadata one. However, they produce similar mean accuracy results (86.3 ± 0.7 % and 86.4 ± 0.2 %) for the complex test dataset. The three rule-based approaches do not reach the precision of any of the others.

Additionally, table 4.4 presents the mean accuracy and mean precision results for the left and right neighbour lanes. The left neighbour lane and right neighbour lane results differ by about 10 % mean accuracy, 20 % mean precision, and 10 % mean F_1score. We assume that this is caused by the unbalanced ratio of existing left and right neighbour lanes in the test datasets. However, the results are promising especially for the right neighbour lane with 90.7 % mean precision for the full test dataset. This might be particularly helpful for predictions of relevant traffic lights, if the ego-vehicle changes suddenly to one of its neighbour lanes.

4.5.5 Investigation of Distances to the Stop Line

We also compared the only-metadata, only-vision and deep metadata fusion approach against human performance with respect to the distance to the stop line of the tested intersections for the traffic light to ego-vehicle lane assignment. We divided the distance to the stop line d into six distance ranges with a constant difference of 15 meters. The results of this investigation are displayed for the full and complex dataset in figure 4.9 together with their 90 % confidence intervals CI.

Our proposed deep metadata fusion approach achieves the best results in general for the full test dataset, see fig. 4.9 (a). It outperforms humans for distances $d \leq 30$ meters. The lower human performance accuracy scores for this distance range are caused by more false negative relevant traffic lights (relevant traffic lights, which are not marked as such).

In figure 4.9 (b), our deep metadata fusion approach achieves significantly higher results than the only-metadata and only-vision approaches for the complex test dataset. However, human performance accuracy is not reached here by our proposed approach. The difference between accuracy scores of human performance and the deep metadata fusion approach is not statistically significant for distances $d \geq 60$ meters with the inclusion of the confidence interval CI.

The only-metadata and only-vision approaches achieve significantly lower accuracy results for all distance ranges in both dataset investigations. In general, this investigation regarding distance ranges confirms our expectation: the closer the ego-vehicle gets to the stop line the more the accuracy increases. It is, however, quite remarkable that our approach can compete and even outperform the humans in many cases.

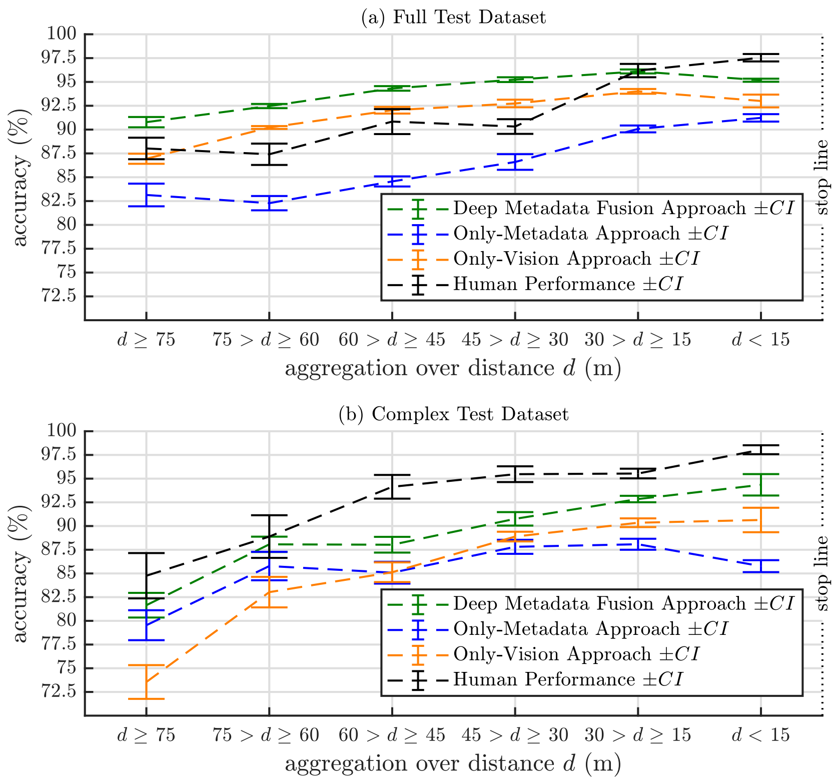


Fig. 4.9 Evaluation over the distance to the stop line for the full (a) and complex test dataset (b) by use of the accuracy metric. Our deep metadata fusion approach (section 4.3) is compared against the only-metadata (subsection 4.4.2), the only-vision approach (subsection 4.4.3), and the human performance (subsection 4.4.4).

4.6 Conclusion

In this work we introduced deep metadata fusion for assigning all relevant traffic lights to their associated lanes.

We trained a common CNN with prepared and transformed input images to predict a vector indicating all relevant traffic light column positions. To achieve this, we fused twelve prepared and adaptively weighted MFMs with convolutional feature maps of a selected convolution layer.

For comparisons, we implemented three rule-based approaches, a machine learning approach, and a deep learning approach. We used for our experiments two test datasets: a complex one, which includes exclusively difficult intersections, and a real world dataset. Both were extracted from a database, which has in total 848 sequences of urban intersection scenarios at daytime. The metadata used was hand-annotated by humans. Hence, we suppose almost perfect metadata quality. In addition, we conducted a subjective test to measure the human performance for assigning all relevant traffic lights to the ego-vehicle lane. It turns out that our deep metadata fusion approach solves the traffic light to lane assignment problem better than other approaches. The baseline for this task is represented by the human performance measured by a subjective test. This baseline is outperformed by our proposed deep metadata fusion approach with +2.7 % mean accuracy for the real world test dataset. The deep metadata fusion technique can also be used to improve other functions like object detection with a RaDAR and camera sensor or lane detection with a LiDAR and camera sensor. In future work, we will develop a sequence-based approach instead of a frame-based approach and test our CNN with metadata similar to real detectors.

Acknowledgement We would like to thank the Daimler AG, which has supported us with the database used. Moreover, we thank our subject group who conducted the subjective test for us.

4.7 Complementary Investigations to the IEEE RA-L Paper

In this section, further experiments are referenced that were not part of the original paper in order to keep the scope focused without additional approach extensions and investigations. Here, the benefit of post-processing, configuration of the deep metadata fusion adaptive weighting, and development of a sequence approach are investigated and concluded.

4.7.1 Benefit of Post-Processing

In both manuscripts, the identical post-processing steps have been applied. The post-processing steps are the output vector matching and the simple majority over frame decision maker, which were introduced in the IEEE ITS paper for the first time, see subsections 3.3.3 and 3.3.4. The benefit of the post-processing steps for the final results is evaluated as follows:

The only-vision and deep metadata fusion approach for the complex and full datasets are evaluated in independent simulations. Each simulation was performed ten times per approach and dataset. Simulations are distinct by random weight initialisation in the convolutional layers. The difference between the final results in and excluding post-processing is calculated for all simulations carried out in table 4.5.

Tab. 4.5 Evaluation of the post-processing benefit by using the mean accuracy over all simulations for both manuscript approaches (cf. chapter 3 and 4) and both datasets.

Approach Name	**Dataset**	**Accuracy** $\pm CI$
Only-Vision Approach	complex	+2.45 ±0.21 %
Only-Vision Approach	full	+2.51 ±0.13 %
Deep Metadata Fusion Approach	complex	+1.72 ±0.16 %
Deep Metadata Fusion Approach	full	+2.20 ±0.09 %

In detail, table 4.5 states the mean difference between the accuracy results with and without post-processing for the only-vision and deep metadata fusion approach as well as for the complex and full dataset. In general, post-processing improves the accuracy results up to 2.5 % with a permanent high reliability noticeable by small 90 % confidence intervals ($CI \leq 0.21$ %). The lowest improvement (+1.72 % accuracy) is achieved for the deep metadata fusion approach with the complex dataset, which is important in order to improve the approach.

In conclusion, all approaches always benefit from post-processing. It reduces the number of flipping single traffic light relevant results by an outlier detection in the smoothed time course of single traffic lights per sequence.

4.7.2 Configuration of Deep Metadata Fusion Adaptive Weighting

Adaptive weighting represents an essential step in the deep metadata fusion approach. It influences MFM weighting by instance normalising each MFM with the global average g^1. Several weighting modifications are configured with higher or lower MFM priorities to determine the optimal configuration.

Tab. 4.6 Investigation of several adaptive weighting configurations by the three mean metric results for the complex test dataset.

No.	**Weighting Modification**	**Accuracy**	**Precision**	**$\mathbf{F_1}$score**
1	adaptive weighting ($g^1 \cdot 0.01$)	86.1 %	83.5 %	82.3 %
2	adaptive weighting ($g^1 \cdot 0.1$)	87.4 %	84.4 %	84.1 %
3	*proposed adaptive weighting ($g^l \cdot 1$)*	*89.2 %*	*87.1 %*	*86.2 %*
4	adaptive weighting ($g^1 \cdot 10$)	88.0 %	85.1 %	84.7 %
5	adaptive weighting ($g^1 \cdot 100$)	87.9 %	85.4 %	84.4 %
6	constant weighting ($g^1 = 1$)	87.9 %	84.6 %	84.9 %
7	constant weighting ($g^1 = 255$)	87.6 %	85.8 %	84.1 %

In table 4.6, all MFM weighting modifications and their mean metric results are listed for the complex test dataset. I used the complex dataset because the deep metadata fusion approach is more successful for this dataset than for the full dataset. This choice allows a more nuanced investigation. MFM weighting modifications No. 1, 2, 4, and 5 use the adaptive weighting explained, cf. subsection 4.3.4. Weights are modified by a scalar multiplication with a factor of different magnitudes. The global average g^1 is multiplied by one hundredth to a hundredfold. MFM weighting modifications No. 6 and 7 represent a constant weighting approach. This means that all MFM values are set to the same constant value for every input image. Weighting modification No. 6 uses logical values ($\{0,\ 1\}$), and weighting modification No. 7 uses RGB colour space values ($\{0, 255\}$).

In contrast to the adaptive weighting approach proposed, see modification No. 3 in table 4.6, all weighting modifications produce lesser results over all three mean metric results. Lower prioritised adaptive weighting modifications No. 1 and 2 achieve the lowest results with an up to 3.9 % lesser $\mathrm{F_1}$score. Higher prioritised adaptive weighting modifications No. 4 and 5 yield similar results to constant weighting modifications No. 6 and 7, with a maximum of 0.8 % difference among themselves for the precision metric.

With regard to the significance of the results, it should be mentioned that the maximum confidence interval range of the complex test dataset is 0.4 %, see table 4.4. Also, the difference between the adaptive weighting proposed and all other weighting modifications lies in the one-digit to four-digit range.

In conclusion, the adaptive weighting global average g^1 regularise most efficiently the fusion of the twelve MFMs by itself, and further modifications of the adaptive weighting are not required in the form of other configurations.

4.7.3 Development of a Sequence Approach

The development of a sequence approach examines the hypothesis that by processing more than one single frame, a higher entropy is generated with regard to the intersection scenario. This hypothesis is inspired by the information recording and processing of humans. It is tested for the deep metadata fusion and the only-vision approach.

The sequence approach is implemented by using an LSTM layer extension, cf. subsection 2.1.3. An LSTM layer is integrated into the proposed CNN topology at three different positions for testing purposes: at the end of the last convolutional layer (Conv. 5), after the first fully connected layer (FC. 1), and after the second fully connected layer (FC. 2), see figure 4.10. Furthermore, the LSTM layer is configured with three iteration cycles: three, five, or seven time steps. This means that the LSTM layer processes the feature inputs from three to seven input images before it produces an output for the following network layer, see figure 4.10. The frame rate is 15 frames per second and on average, every third frame is annotated. Thus, the shortest sequence time of an iteration cycle is 0.4 seconds (three time steps), and the longest time is 1.4 seconds (seven time steps).

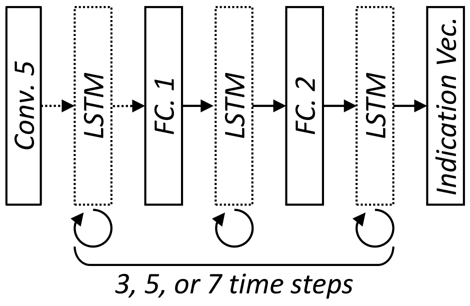


Fig. 4.10 Sequence approach with an LSTM layer extension. Three different integration positions and three iteration cycles of the LSTM layer are investigated.

The parameter setup for the sequence approach is the same as for the deep metadata fusion[33] or only-vision approach. Nevertheless, the CNN of the sequence approach is trained in a two step procedure. First, the CNN is trained without any LSTM layer as described in subsection 4.3.5. Second, the layer outputs of the neural network layers before the respective LSTM layer (Conv. 5, FC. 1, and FC. 2) are stored separately. These layer outputs are used to build a shortened neural network, which starts with an LSTM layer and ends with the indication vector output. This neural network is trained again for up to 100 epochs with the layer outputs as LSTM layer inputs for all three integration positions and iteration cycles. This procedure saves training run-time

[33]Here, the convolutional layer two (Conv. 2) was used as a fusion layer for the deep metadata fusion approach, which was proposed as the one of the most successful fusion layers, cf. subsection 4.5.2.

and ensures a correct comparison between the deep metadata fusion and only-vision approach with and without an LSTM layer extension.

The mean accuracy results for the sequence approach are displayed in figure 4.11. The figure shows the results regarding the deep metadata fusion approach and only-vision approach, the full test dataset and complex one, all three LSTM layer integration positions, and all three iteration cycles in comparison to the results without an LSTM layer extension. The LSTM layer extension does not reach any significant mean accuracy improvements for the deep metadata fusion approach with the complex test dataset. However, the only-vision approach achieves up to 1.2 % more accuracy for all LSTM layer integration positions and iteration cycles than the only-vision approach without the LSTM layer extension with the complex test dataset. The deep metadata fusion and only-vision approaches with LSTM layer extension do not outperform their baselines

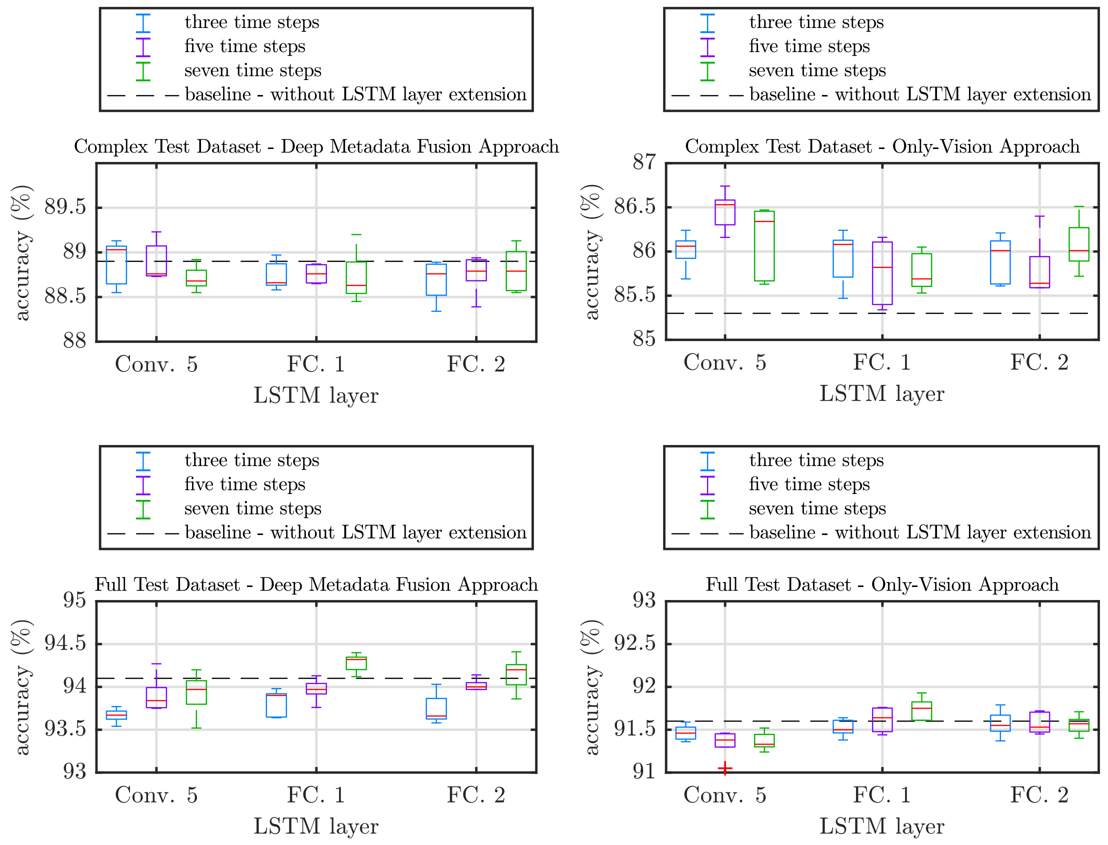


Fig. 4.11 Sequence approach results with an LSTM layer extension. The LSTM layer extension is applied to the only-vision and deep metadata fusion approach for the complex and full test datasets. The LSTM layer extension for the only-vision approach with the complex test dataset outperforms the baseline for all integration positions (Conv. 5 to FC. 2) and iteration cycles (three to seven time steps).

with one exception by using the full test dataset. This exception applies for the LSTM layer integration after the first fully connected layer (FC. 1) with seven time steps, cf. figure 4.11. Here, both approaches significantly outperform their baseline with up to 0.4 % mean accuracy. It is assumed that these are two outliers and the better results are also possible without the LSTM layer extension.

Note that the evaluation results of the precision and F_1score metrics are relative to the visualised accuracy results, and for other absolute value ranges, see appendix section 7.7, where all results are listed.

In conclusion, the LSTM layer extension improves the results of the only-vision approach with the complex test dataset for all integration positions and integration cycles. This is due to the sequence approach, which generates new useful features over-time to resolve the TL2LA problem. By contrast, the LSTM layer extension does not take advantage of the over-time generated features for the deep metadata fusion approach. It is assumed that no more improvements are possible over-time for the deep metadata fusion approach with both datasets because the approach is already saturated through the use of the encoded features in the twelve MFMs.

4.8 Brief Discussion of the IEEE RA-L Paper

In this section, specific implementation details of the IEEE RA-L paper on the deep metadata fusion approach are briefly discussed: the MFM preparation, the deep metadata fusion layer and fusion operator, and the impact of single MFMs. The approach in its entirety and its importance for further applications and research will be discussed in chapter 5.

The MFM preparation was explained in subsection 4.3.3. During this pre-processing, the heterogeneous data sources are harmonised for the deep metadata fusion approach. The various annotated metadata objects (traffic lights, lane line markings, lane arrow markings, and lane signs) are transferred into twelve binary-valued images of the same size – the MFMs. The correct scale and mapping between the annotated metadata objects and the generated MFMs must be used, otherwise, the deep metadata fusion will not work due to incorrect metadata feature positions.

The fusion operator of the deep metadata fusion approach was investigated in comparison to two others, cf. subsection 4.5.2: element-wise addition, vanilla concatenation and element-wise multiplication (proposed one). It turns out that the proposed element-wise multiplication fusion operator performs best due to the ability to scale features of an MFM with the previously generated convolutional features, cf. table 4.4 in subsection 4.5.4. In contrast to the feature scale of the element-wise multiplication operator, the element-wise addition does a feature shift which can be partly compensated by the normalisation layers. The concatenation is similar to the element-wise addition because the actual fusion takes place in the following convolution layer. There, the MFMs and the convolutional feature maps are convolved by a single kernel and subsequently element-wise added in order to generate a new convolutional feature map.

The fusion layer was investigated in subsection 4.5.2. It was found that an early fusion (in Conv. 1 to Conv. 3) is more successful because the MFMs have to be down-sampled too much for a later fusion (in Conv. 4 or Conv. 5). The encoded metadata information is therefore largely lost. In addition, there is a growing inequality between fused MFMs and unfused (passed-through) convolutional feature maps. For example, twelve fused MFMs are confronted with 116 unfused convolutional feature maps in Conv. 4. This creates a lower relevance of the fused MFMs in the CNN.

The effect of each MFM was investigated with an ablation study in subsection 4.5.3. It was found that eleven out of twelve MFMs have a positive effect on the results between +1.1 % and +2.2 % accuracy, cf. figure 4.8. The highest effect is attributed to MFM No. 1 (all traffic lights) and No. 8 (ego-vehicle lane line markings) because these MFMs contain the most important features to resolve the TL2LA problem. The MFM No. 6 (traffic lights with right direction arrow) has no significant effect on the results with +0.2 % accuracy. It is assumed that this is because of the inner dataset distribution. Traffic lights with right direction arrows are underrepresented in comparison to traffic lights with left and straight direction arrows (MFM No. 4 and 5) conditioned by intersections in Germany. In total, all MFMs concerning traffic light and lane metadata would have a deep metadata fusion effect of about +12.5 % accuracy (addition of all single MFM impacts, cf. figure 4.8). However, the individual MFM results of the ablation study cannot be added together because they are not distinct from each other. What results is a lesser total deep metadata fusion effect[34] of 2.9 % greater accuracy, +4.4 % more precision, and +4.1 % more F_1score as taken from table 4.4 in subsection 4.5.4.

[34]Absolute difference between the only-vision and deep metadata fusion approach for the complex test dataset.

5 Discussion

In the following, five cross-sectional aspects of this dissertation are discussed. These aspects are the database, the IPM method, the subjective tests, the deep metadata fusion approach, and the TL2LA.

Database The database used is discussed concerning the quality and usability of the object annotations. A discussion about the dataset size was already made in subsection 3.6.2 for the IEEE ITS paper.

The quality of the annotated metadata objects, cf. figure 7.1 in the appendix, is affected by random human annotation errors and predefined annotation margins. The metadata used, such as lane arrow markings, lane line markings, and lane signs are annotated with a margin of ± 5 pixels in the horizontal and vertical object directions. The traffic light metadata used were annotated with a margin of ± 2 pixels. These annotation margins are satisfactory in order to use the metadata objects in the form of MFMs for the deep metadata fusion approach because the annotation margin error becomes smaller together with the up to factor eight down-sampled MFMs. Furthermore, the annotation margin of the traffic lights is exact enough to use these metadata for a detection task, what it were originally intended for, cf. the DriveU dataset [12].

Moreover, the usage of annotated metadata raises the following research question concerning the usage of metadata from real detectors: How much do the accuracy, precision, and F_1score metrics decrease by using different false positive (FP) and false negative (FN) ratios of the metadata objects for the deep metadata fusion approach? FPs and FNs of the annotated metadata objects are currently very rare and quantitatively not measurable without a validation of each metadata object. In this work, it is assumed that the annotated metadata does not have any FPs or FNs. However, the creation of metadata from real detectors would be an independent research project and could be part of a subsequent project of this thesis. It could be created by (1) generating of FPs and FNs into the database of the annotated metadata objects with a random object generator or by (2) capturing FPs and FNs metadata objects by means of algorithms from real detectors which are fed with the recorded camera images. At the moment, this research question cannot be answered exactly. Nevertheless, the question regarding metadata from real detectors would not arise anymore if real detectors achieved detection rates[35] close to 100 % as is necessary for level three driving or higher.

IPM method The method to generate top view images from camera images was explained in section 2.2. It is used to prepare the input images for the only-vision and deep metadata fusion approach. The main contribution in comparison to other state-of-the-art IPM transformations is that this IPM method is independent from extrinsic camera parameters such as the pitch angle. Moreover, the method allows to create an IPM full panorama image, cf. figure 2.9, that is used to annotate all lane line markings with less time effort and minor lateral errors up to medium distances of 50 m, cf. figure 2.12. This IPM lane line marking annotation principle would have the potential to replace the standard annotation procedure if minimal lateral errors were acceptable or further improvements followed.

A disadvantage of this IPM method is that it is currently not real-time capable, because it needs a lot of computing power for the RANSAC algorithm and the MAE optimisation to determine the road plane and the image horizon. A further improvement could be, e.g. a semantic segmentation of the free space in the camera image. Then only relevant three-dimensional road plane points are passed to the RANSAC algorithm to improve the road plane estimation, cf subsection 2.2.1, and reduce the processing time of the algorithm.

Subjective Tests Two subjective tests were conducted in this work. Both subjective tests had the same goal: Subjects should identify all relevant traffic lights for the ego-vehicle lane.

The first one was designed by me to verify the performance of the only-vision approach in the IEEE ITS paper, see subsection 3.4.4. The subjects were shown the prepared CNN input images with one difference: The images were up-scaled from (256, 256, 3) to (512, 512, 3) RGB pixels in height and width to obtain more convenient handling. Hence, the first subjective test design has three disadvantages:

1. The subjects had to interpret the top view of the road part in the input images. This is an unusual perspective for humans and could produce some misleading information by the identification of the ego-vehicle lane.

2. The image resolution was relatively low. This led to smaller traffic lights, which are far away from the ego-vehicle and more difficult to observe as well as to mark with the mouse cursor. Moreover, traffic light icons, e.g. direction arrows, had

[35]This is measured by the true-positive-rate or precision metric.

a low resolution. Thus, they were not clearly visible for the subjects in the test images.

3. The subjects were always shown a single input image like the CNN also processes and evaluates. But the assessment of one single image is also unusual for humans, who are more familiar with videos or image sequences.

However, the test should not measure the real human performance. It is a methodical comparison to the CNN results of the only-vision approach, since other approaches for comparison, except the rule-based ones, are not available in the reference literature.

The second subjective test, see subsection 4.4.4 in the IEEE RA-L paper, addresses the three disadvantages of the first one. It uses the originally recorded camera images, a high resolution with (1792, 896, 3) RGB pixels, and plays a three-second long video of the intersection scenario before the actual assessment of the subjects starts. However, one bottleneck remains in both subjective test designs: Each subject requires more or less time to mark all relevant traffic lights by means of the mouse cursor. This delay in handling means that the ten-second assessment or processing time was not enough in exceptional cases.

Moreover, the second subjective test had the aim to measure the real human performance regarding the TL2LA problem. Human performance is outperformed by the proposed deep metadata fusion approach for the full test dataset in the accuracy and F_1score metrics with up to 2.7 %. This raises the research question as to whether the deep metadata fusion approach solves the TL2LA problem more successfully than humans. The answer can be determined by using the precision metric. This metric describes the true-positive-rate of relevant traffic lights for the ego-vehicle lane and is independent from the number of marked as relevant traffic lights by the subjects. The precision of the subjective test always outperforms all other approaches, see subsection 4.5.4. The only exception is 92.9 $\pm$1.3 % (human performance) to 93.0 $\pm$0.2 % (deep metadata fusion approach) for the full test dataset, but this is not significant. Thus, the better accuracy and F_1score results are caused by subjects who have marked only one instead of all relevant traffic lights for the ego-vehicle lane per test sequence. In conclusion, the deep metadata fusion approach is almost as accurate as the measured human performance with the second subjective test.

Deep Metadata Fusion Approach The development of the deep metadata fusion approach[36] to resolve the TL2LA problem represents the main contribution of this thesis. The approach is based on the encoding of relevant traffic light column positions

[36]The approach is based on a combination of existing neural network layer types like merge, multiplication, and normalisation layers which are available in open-source frameworks, cf. [43], [80].

in an indication vector. This was introduced with the only-vision approach in the IEEE ITS paper for the first time, cf. chapter 3, and enables a CNN to transfer a one-to-many assignment problem into a regression problem.

For comparison, other approaches were explained and tested to resolve the TL2LA problem. Diverse rule-based approaches (traffic light mapping, main traffic light, and traffic light above ego-vehicle lane, cf. subsection 4.4.1), an only-metadata approach[37], cf. subsection 4.4.2, and the only-vision approach, cf. subsection 4.4.3, were evaluated. Moreover, an error analysis revealed that all these approaches have a joint overlap of 5.0 % accuracy errors, details are explained in section 7.6 in the appendix. In sum, none of these approaches were able to achieve similar or even better results like the deep metadata fusion approach, cf. table 4.4 in subsection 4.5.4. It was also shown that this approach can be applied to more than one lane, e.g. the left and right neighbouring lanes.

The deep metadata fusion approach was investigated for various optimisations: three different fusion operators (cf. subsection 4.5.2), several adaptive weighting modifications (cf. subsection 4.7.2), an LSTM layer extension (cf. subsection 4.7.3), and the effect of each MFM on the results (cf. subsection 4.5.3). In conclusion, the most efficient deep metadata fusion approach configuration uses an element-wise multiplication fusion operator, an unmodified global average scalar for the adaptive weighting, and all twelve available MFMs. In contrast to the only-vision approach, which benefits from an LSTM layer extension by using over-time generated features, the deep metadata fusion approach cannot be improved with an LSTM layer extension.

The main advantages of this novel deep metadata fusion approach are as follows in comparison to other deep fusion approaches:

- It is fast in forward propagation, especially the adaptive weighting technique because it uses exclusively the global average of all previous convolutional layers to weight all MFMs instead of additional learned scale and shift parameters like they are learned for layer normalisation, cf. subsection 2.1.5.
- The deep metadata fusion approach works without additional backward propagation. After the preparation of the MFMs, the MFMs are immediately fused into the CNN without prefixed convolutional layers as is implemented in other deep fusion approaches, e.g. in [76], [60], and [79], which requires to set up more hyper parameters and a longer training run-time.
- In a direct comparison, the deep metadata fusion approach demonstrates that a more complex fusion operator like the proposed element-wise multiplication

[37]This approach is based on the annotated metadata and traffic lights, which are packed into a feature vector. The feature vector is described in more detail in table 7.1 in the appendix.

leads to better results than a simpler fusion operator like concatenation (with up to +2.0 % accuracy, 2.9 % precision, and +1.8 % F_1score, cf. table 4.4). This could influence further investigations regarding the deep fusion in CNNs, cf. subsection 2.1.6, because most deep fusion approaches use aggregation fusion (maximisation, minimisation, summation, or concatenation), e.g. in [81], [57], and [79].

A disadvantage of the deep metadata fusion approach as well as for the only-vision one is that traffic lights which hang directly above each other cannot be differentiated with the indication vector in rare cases. Here, an indication matrix instead of an indication vector would compensate this disadvantage, but it should be ensured that the additional row dimension does not have too many elements, e.g. an indication matrix size of (2, 256) values would be sufficient for an upper and a lower horizontal level of traffic lights. Otherwise, it could lead to training complications with the loss function. If the proportion of unset (zeros) to set values in the indication matrix was very unbalanced, the CNN would learn the simplest solution that all indication matrix values are always zero.

Furthermore, optimisation could be included to add additional MFMs to the deep metadata fusion approach, which contain, e.g. depth maps [58], [59] or semantically segmented images [79]. This would extend the deep metadata fusion approach by a raw level fusion and could focus the CNN on more important features for the TL2LA function. Moreover, an extension of the CNN topology implementation with a capsuled neural network layer [82] could be validated. This layer has the property to decide between different feature hierarchies by bringing them into a logical order, which could be beneficial for the TL2LA.

Traffic Light to Lane Assignment (TL2LA) The TL2LA function was the motivator behind the development of the deep metadata fusion approach in this thesis. In addition, the development of a TL2LA function could also be resolved with other approaches than the made only-vision or deep metadata fusion one. In the following, three other approach options are discussed:

- A combination of simpler approaches can be implemented. For example, the results of the only-vision approach can be used as additional feature vector input for the only-metadata approach. This would then be a composition of two approaches. This special composition was also validated, but has not achieved significantly better results than the only-vision approach, see section 7.5 in the appendix for more details.

- The TL2LA function can be realised with traffic infrastructure data, e.g. HD maps provided by a cloud data management system. Traffic light state, position, and relevance information are transmitted via I2V to the vehicle. Some I2V traffic light information concepts are already proposed in [68] and [67]. In addition, a simulation study with traffic infrastructure data to decide how fast a vehicle should drive towards the relevant traffic light was conducted in [69]. However, such HD maps provided by I2V are only available for small regions, are error-prone for city construction, and are expensive to create at the moment. Moreover, the expansion of the I2V infrastructure, e.g. with the mobile network 5G, is also not finished yet.
- The development of a TL2LA function can be simplified by an equal traffic light state restriction. This means that the vehicle will brake on a stop line if all traffic light states are red. If all traffic light states are green, the vehicles will drive through the intersection. The advantage of this restriction is that the TL2LA function would be very reliable. However, it is obvious that many cases will be not addressed with this restriction, particularly complex intersection scenarios.

In conclusion, the deep metadata fusion approach for the TL2LA function represents an approach option that can be implemented immediately, is cost effective, and dominates the majority of intersection scenarios at the moment.

6 Summary and Outlook

Summary This work dealt with a novel deep metadata fusion approach to resolve the many-to-many traffic light to lane assignment (TL2LA) problem. The work closed the research gap between Traffic Light Recognition (TLR) and Traffic Light Assistance (TLA) systems. It enabled TLA systems to interpret recognised traffic lights in the right context by assigning them to their relevant lanes. The deep metadata fusion approach was developed by means of a database with 848 real intersection scenarios and over 45 thousand frames, which were recorded by a stereo camera. Each frame was annotated by humans with traffic lights, lane line markings, lane arrow markings, and lane signs as metadata objects, cf. chapter 1.

In chapter 2, the theory regarding deep learning and deep fusion with Convolutional Neural Networks (CNN) was explained. The presented deep metadata fusion approach was categorised as a direct deep feature fusion approach with heterogeneous data streams. Moreover, an Inverse Perspective Mapping (IPM) method was developed in order to transform camera images into top view images as well as to annotate efficient lane line markings by use of an IPM full panorama image of the intersection scenario or sequence. The IPM method was developed independently from extrinsic camera parameters such as the camera pitch angle. It used a flat world estimation of the road part in a three-dimensional point cloud image by use of the RANSAC algorithm. Subsequently, it compared the top view image against the previously created top view image of the sequence by use of an Mean Absolute Error (MAE) to optimise the IPM transformation.

In the first manuscript, cf. chapter 3, an only-vision approach was developed. This approach resolved the TL2LA problem by vision only. The main contributions of the only-vision approach were: (1) The road part of the input images was transformed into a top view image by use of the IPM method. This conditioned more symmetry between different CNN input images and improves the results. (2) As CNN output, an output indication vector was defined. This indication vector enabled the CNN to transfer the assignment problem into a regression problem. For that the indication vector encoded the column positions of all relevant traffic lights in the input image. As result, the only-vision approaches achieved 86.4 % average accuracy. In addition, the

CNN topology, dataset size and CNN input image size were investigated. In conclusion, an AlexNet CNN topology resolved the TL2LA problem with high accuracy results and a low 90 % confidence interval over many simulations by use of an acceptable training run-time. An enlargement of the available dataset size or the chosen input image size did not achieve higher accuracy results.

In the second manuscript, cf. chapter 4, the deep metadata fusion approach was developed. It was based on the processing steps and the contribution of the only-vision approach. Furthermore, the deep metadata fusion approach encoded the metadata objects into twelve binary Metadata Feature Maps (MFM) of the same size as the selected convolutional fusion layer. These MFMs were fused by use of an adaptive weighting technique and an element-wise multiplication fusion operator into the selected convolutional fusion layer. The adaptive weighting technique used the global average of the previous convolutional layer. It was discovered that an element-wise multiplication as fusion operator, an early fusion in convolutional layer one to three, and the use of all metadata feature maps achieve the highest results with 93.7 % average accuracy for the full dataset. The deep metadata fusion approach was compared against three rule-based, an only-metadata, and an only-vision approach. It outperformed significantly all other approaches and the results increased the closer the vehicle approaches the stop line up to 96.7 % accuracy for distances between 15 m to 30 m for the full dataset. However, traffic lights which are positioned close together but are not relevant for the same lanes are still a challenge for the deep metadata fusion approach. In addition, a subjective test was conducted to measure human performance and to obtain a baseline for the TL2LA problem. For this, a subjective test tool was programmed and rolled out to a heterogeneous group of 40 subjects. The subjective test design was based on a previous performed subjective test and remedied its disadvantages because the previous subjective test was conducted to evaluate the CNN model performance of the only-vision approach. In conclusion, the human performance and deep metadata fusion approach achieved almost identical precision (true-positive-rate) results. However, the deep metadata fusion approach outperformed the human performance in the accuracy metric for the full test dataset with +2.7 %. Moreover, the only-vision approach and deep metadata fusion approach were extended to a sequence approach by use of an Long Short Term Memory (LSTM) layer. The result was that the only-vision approach benefited exclusively from this extension.

In the last chapter 5, the quality of the annotated metadata objects and the usage of metadata from real detectors were discussed. The disadvantages of the first subjective test and their compensation with the second one were discussed and the results of the measured human performance and the deep metadata fusion approach were interpreted. In addition, the advantages and disadvantages of the deep metadata fusion approach as well as possible optimisations of the approach were mentioned. Finally, three alternative approaches to create a TL2LA function were suggested.

Moreover, a video[38] was created that exemplifies the traffic light to lane assignment, the deep metadata fusion approach, and the second subjective test.

Outlook The deep metadata fusion approach could be adapted to other use cases in the field of ADAS. It could be resolved for further assignment problems, e.g. vehicle to lane assignment. In the reference literature, the preceding vehicle to lane assignment problem is resolved by rule-based, filter-based, or Multi-Layer-Perceptron (MLP) approaches. For example, it is checked as to whether the preceding vehicle is in the ego-vehicle-lane, cf. [83], a Bayes filter generates a discrete posterior probability for a preceding vehicle in the ego-vehicle-lane, cf. [84], or an MLP is trained with lane and vehicle features in order to decide on which lane index the preceding vehicles are located, cf. [85]. The deep metadata fusion approach could be used to resolve this assignment problem by using the same CNN input images and use case specific MFMs.

The deep metadata fusion approach could also be used for deep feature fusion of different senor data sources, RaDar, camera, or LiDar, to detect or classify objects. For example, object detection with camera and LiDar sensor data uses region based or gated based deep fusion approaches in the reference literature. Region based deep fusion processes separately the camera and LiDar data and fuses their object proposals together by using an element-wise mean fusion operator, cf. [86]. Deep gated fusion fuses generated camera and LiDar feature maps pixel-wise together depending on the gated fusion unit decision after each convolutional layer, cf. [87].

Moreover, the deep metadata fusion approach could be extended to a full end-to-end re-enforcement learner by implementing it in a vehicle fleet. This would open up the opportunity to use a large amount of training data via cloud data management and to train a global CNN that has local copies in each vehicle. The sensor input, camera images and metadata, would be stay the same. The actor output would be the information as to whether a vehicle should brake on or run over a stop line at a signalised intersection instead of the output indication vector for relevant and non-relevant traffic lights. However, drivers who do not follow the traffic rules can compromise this approach extension.

I created my dissertation in the form of an offline development. The next step should be to integrate the deep metadata fusion approach into a test vehicle to validate the approach in real world. This test vehicle must have a TLR and Lane Detection (LD) function to create the required MFMs.

[38]Link to the video of the IEEE RA-L paper [7]: https://ieeexplore.ieee.org/ielx7/7083369/8581687/8613841/ieee_ra_l_video.mp4?tp=&arnumber=8613841.

7 Appendix

7.1 Database Annotations of one Example Image

In this section, the database annotations which were explained in chapter 1 are visualised based on an example image.

Figure 7.1 shows an example image with all objects annotated by humans. Traffic lights, lane arrow markings, and lane signs were annotated as rectangles. The lane line markings were annotated as polylines in the IPM full panorama image and re-transformed into the left camera image, cf. subsection 2.10. The traffic light to lane connections (ground truth) were created by using an allocation matrix with predefined identification numbers of each lane and traffic light. An allocation matrix was filled for each database frame by hand, cf. equation 1.1 that shows an example allocation matrix.

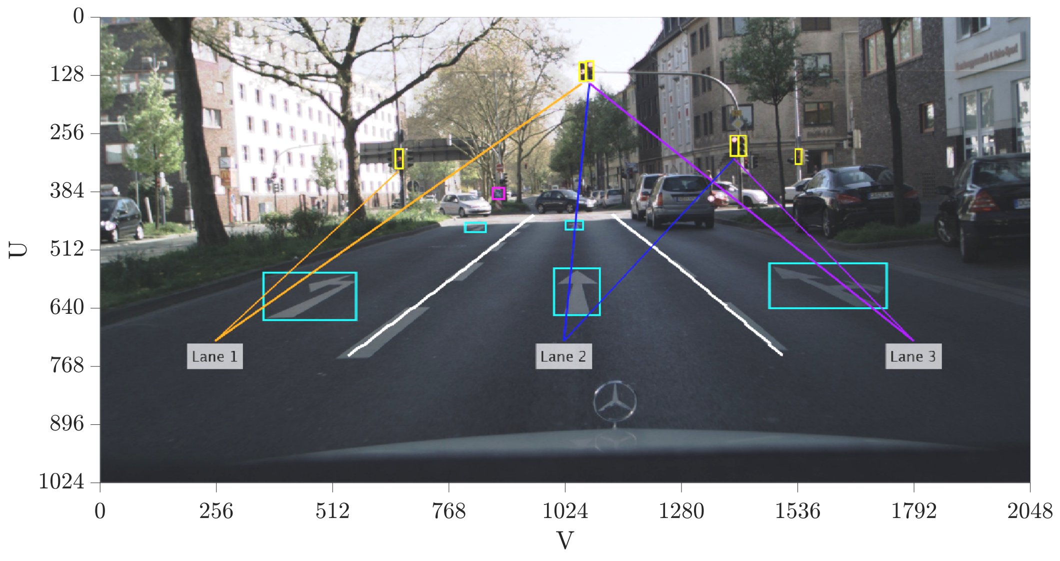


Fig. 7.1 Example of one annotated image: Traffic lights (yellow), lane line markings (white), lane arrow markings (cyan), and lane signs (magenta) are annotated. The TL2LA ground truth (orange, blue, and purple traffic light to lane connections) are shown for each lane.

7.2 Example of an IPM Lane Line Marking Image Evaluation

In this section, a single IPM lane line marking image is evaluated. This image corresponds to the example image in figure 2.10 (a), cf. subsection 2.2.5.

Figure 7.2 shows the lateral error over the longitudinal distance of the IPM lane line markings. The lateral error has its maximum at 48 m with an MAE of 18 cm. The lateral error of 5 cm can be interpreted as a default error at the beginning of the IPM lane line marking annotation (10 m). This default lateral error is caused by off-centred annotations of the lane line markings. A real world lane line marking has a width of about 12 cm or six pixels (1 px = 0.02 m) in the IPM full panorama image. Thus, this example image has an annotation margin of about 2.5 pixels.

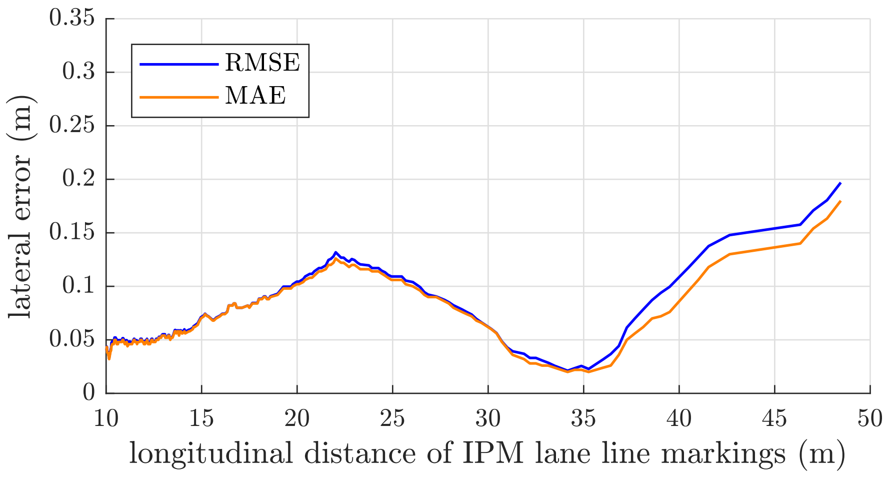


Fig. 7.2 Evaluation of the IPM lane line marking exactness for a single image through calculation of the lateral error by using reference lane line markings for the ego-vehicle lane. The RMSE and MAE have their maxima at 48 m and the default lateral error is about 5 cm.

7.3 Feature Vector Attributes of the Only-Metadata Approach

This section provides additional information on the feature vector attributes of the only-metadata approach. Table 7.1 explains the encoded feature information of the input feature vector for the only-metadata approach. It is similar to table 4.2 in subsection 4.4.2. However, table 7.1 contains an additional column for the value range to describe the 31 features in more detail. This column was not added to the original manuscript of the IEEE RA-L paper because of space restrictions, cf. chapter 4.

Tab. 7.1 Feature list of the only-metadata approach including the value range and the encoding for the input feature vector.

No.	Description	Value Range
1-3	lateral, longitudinal, and vertical traffic light position	[0.1, 128.0] meter
4	traffic light colour (unknown, red, yellow, green, red-yellow) coding	[0, 4] binary
5-6	global traffic light height and width	[0.1, 1.5] meter
7-9	mean lateral, longitudinal, and vertical traffic light assembly position	[0.1, 128.0] meter
10-12	traffic light left, straight, and right direction arrow type coding	{0, 1} binary
13-15	current lane left, straight, and right direction type coding	{0, 1} binary
16-17	number of left and right neighbour lanes	[0, 5] number
18-19	left-sided and right-sided lateral ego-vehicle lane line marking position below traffic light assembly	[-32.0, 32.0] meter
20-22	nearest lateral, longitudinal, and vertical lane sign position	[0.1, 128.0] meter
23-25	nearest lane sign left, straight, and right direction type coding	{0, 1} binary
26-28	nearest lateral, longitudinal, and vertical lane arrow position	[0.1, 128.0] meter
29-31	nearest lane arrow left, straight, and right direction type coding	{0, 1} binary

7.4 CNN Output Visualisation of the Only-Vision Approach

In this section, the entire output of the only-vision approach is visualised and described to explain the functionality of the output regression vector in more detail.

Figure 7.3 displays the output of the CNN model or only-vision approach for the complex test dataset. The approach was described in the IEEE ITS paper, cf. chapter 3. The figure shows all target regression vectors, the output regression vectors, and the matching vectors in the form of a binary image for each test frame of a sequence. The three regression vectors and the calculation of the traffic light relevant results are described in subsection 3.3.3. Note that these vectors are renamed as indication vectors in the IEEE RA-L paper, cf. chapter 4. The matching vector contains all traffic lights (relevant and non-relevant traffic lights for the ego-vehicle lane) of the complex test dataset. The output regression vector contains the predicted relevant traffic lights for the ego-vehicle, and the target regression vector represents the ground truth for all relevant traffic lights. Qualitative results of the only-vision approach can be examined by comparing the target and output regression vectors. For example, in sequence 13 the traffic light on the right side (at vector length 224 to 256 pixels) is assigned as an

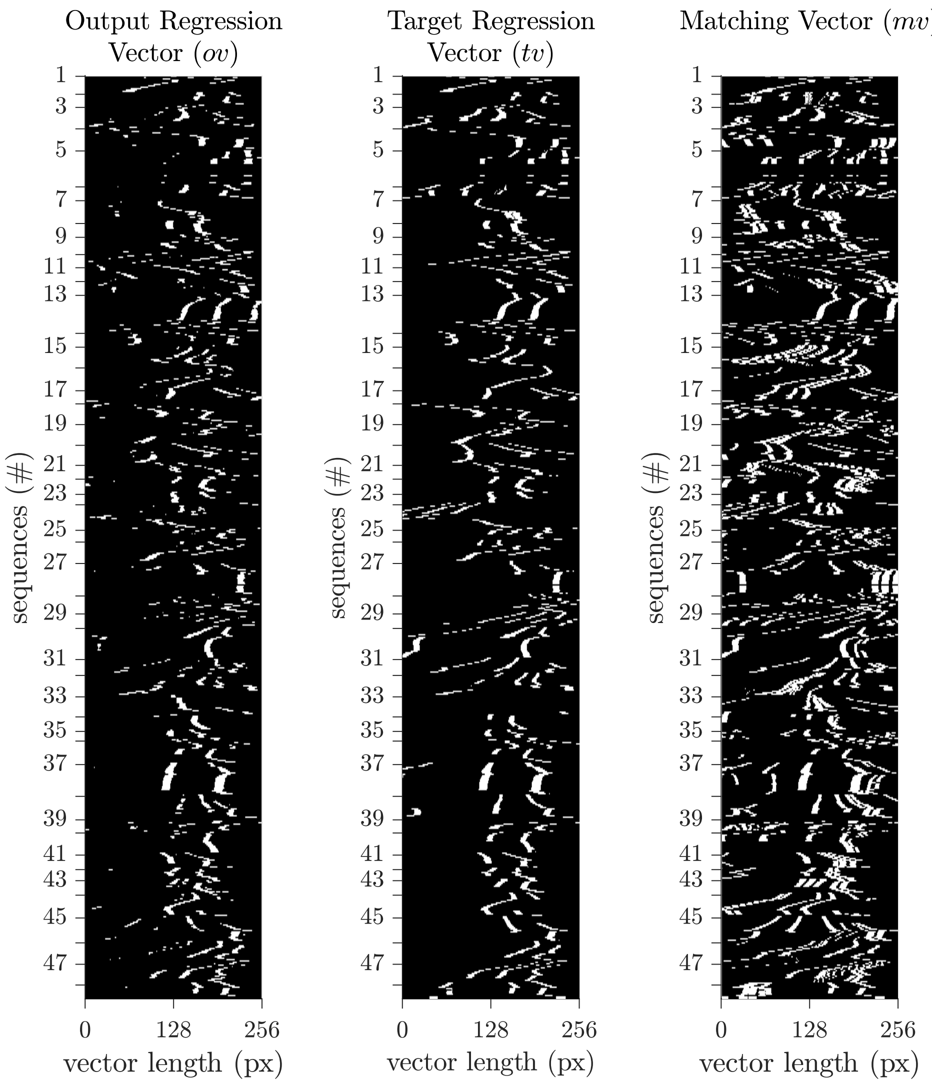


Fig. 7.3 Qualitative results of the only-vision approach. The output regression vector contains the predicted relevant traffic lights for the ego-vehicle lane. The target regression vector contains the ground truth. The matching vector contains all (relevant and non-relevant) traffic lights.

FP. In sequence 38, the traffic light on the left side (at vector length 0 to 32 pixels) is assigned as an FN.

7.5 Only-Vision and Only-Metadata Approach Composition

In this section, a composition of the only-vision and only-metadata approach was made. The idea was that the only-metadata approach uses the output of the only-vision approach as additional input.

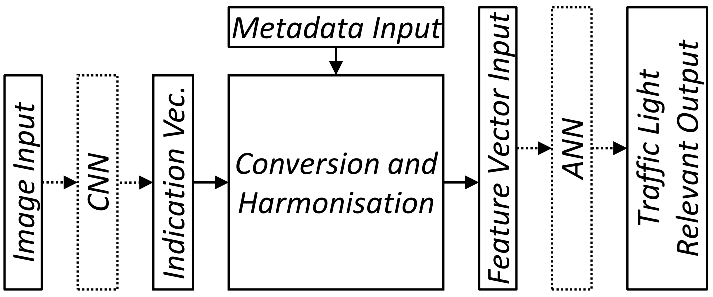


Fig. 7.4 Composition of the only-vision and only-metadata approaches. The feature vector input for the only-metadata approach is extended by one additional feature. This additional feature is the binary encoded traffic light relevant results of the only-vision approach.

Figure 7.4 explains the composition of both approaches by means of conversion and harmonisation of their input and output data. The only-vision approach and the only-metadata approach are executed as described in subsections 4.4.2 and 4.4.3. In addition, the feature vector input is extended by the traffic light relevant results of the only-vision approach after post-processing, cf. subsection 3.3.3. The traffic light relevant results are encoded as a binary feature: a one for a relevant and a zero for a non-relevant traffic light. Thus, the resulting feature vector input for the ANN of the only-metadata approach has 32 instead of 31 features, cf. table 7.1.

Table 7.2 displays the results of the approach composition as well as the results for every single approach taken from table 4.4 and figure 4.9 of the IEEE RA-L paper with the complex test dataset. The approach composition results can be understood as a maximum function between the only-vision and only-metadata approaches. This can

Tab. 7.2 Results of the only-vision and only-metadata approach composition for the complex test dataset.

Approach	**Mean Metrics**			**Accuracy over Distances d**					
	Accuracy	Precision	F_1score	$d \geq 75$	$75 < d \geq 60$	$60 < d \geq 45$	$45 < d \geq 30$	$30 < d \geq 15$	$d < 15$
Composition[1]	87.5 %	83.0 %	82.4 %	80.2 %	84.1 %	85.2 %	89.0 %	90.6 %	91.0 %
Only-Vision[1]	86.4 %	82.7 %	82.1 %	73.6 %	83.0 %	85.1 %	88.9 %	90.4 %	90.6 %
Only-Metadata[1]	86.3 %	77.2 %	82.0 %	79.5 %	85.8 %	85.0 %	87.8 %	88.1 %	85.7 %

[1]The minimum 90 % confidence interval for all accuracy over distance results is ±0.6 %.

be seen by using the accuracy over distance evaluation in table 7.2. The only-vision approach already outperforms the only-metadata approach for most distances to the top line. One exception is that the only-metadata approach achieves an accuracy that is +2.8 % greater in the distance range between 60 m and 75 m.

In conclusion, the composition of these two approaches achieves a 1.1 % better mean accuracy than the only-vision approach, but the results do not match those of the deep metadata fusion approach with a mean accuracy of 89.3 %, cf. table 4.4. Thus, this composition of both approaches is not an alternative to the deep metadata fusion approach proposed in chapter 4.

7.6 Error Analysis of Different IEEE RA-L Paper Approaches

In this section, an error analysis is performed for the proposed deep metadata fusion approach as well as for the other approaches that were used for comparison in the IEEE RA-L paper.

Figure 7.5 shows the error overlaps of the only-metadata, only-vision, and deep metadata fusion approaches. The areas of the circles visualise the mean accuracy error for each approach. Error quantities[39] are taken from table 4.4 for the full test dataset. Error overlaps between the three approaches are calculated by comparing the single test frame results with each other.

The accuracy error overlap between the only-metadata and only-vision approach is 6.7 % and the accuracy error overlap between all three approaches is 5.0 %. The only-vision and deep metadata fusion approach have the highest error overlap. There are 0.2 % accuracy errors of the only-vision approach outside of the deep metadata fusion approach error quantity, see figure 7.5.

In conclusion, 5.0 % of full test dataset frames are generated with accuracy errors by all approaches. This means that these test frames have at least one TN or FN relevant traffic light. Moreover, the three rule-based approaches are not analysed in detail because they are exclusively measurable in the precision metric. The TNs of these three rule-based approaches are superimposed with the only-metadata, only-vision, and deep metadata fusion approach error quantities. Furthermore, the error quantities of the human performance measured in the (second[40]) subjective test are also not analysed because they are based on a random sample of the full test dataset. However, it was found that each single test frame of the subjective test was assessed correctly by at least one subject.

[39]Error quantities are calculated by 100 % minus the mean accuracy results.

[40]This subjective test is described in subsection 4.4.4 in the IEEE RA-L paper, cf. chapter 4.

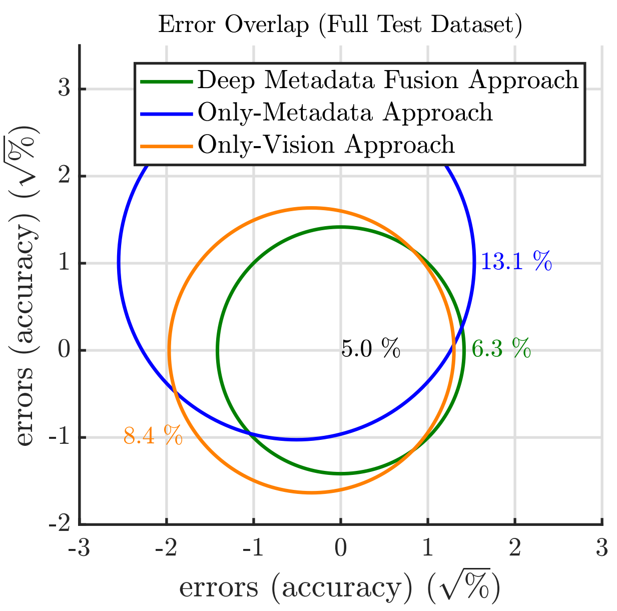


Fig. 7.5 Intersection diagram of the error overlap in the accuracy metric for the only-metadata, only-vision, and deep metadata fusion approaches with the full test dataset. The highest error overlap is exhibited by the only-vision and deep metadata fusion approach. All three approaches have a joint overlap of 5.0 % accuracy errors.

7.7 Experimental Results of the Sequence Approach

In this section, all simulation results of the LSTM layer extension for the only-vision and deep metadata fusion approaches are listed, cf. subsection 4.7.3.

Tables 7.3, 7.4, 7.5, and 7.6 represent the mean accuracy, precision, and F_1score as well as the accuracy results regarding the distance to the stop line for each simulation. In total, 45 simulation were run (5 repetitions, 3 integration positions, and 3 iteration cycles) for each approach (only-vision and deep metadata fusion approach) and dataset (complex and full dataset).

The results in tables 7.3 and 7.5 show that the deep metadata fusion approach is unresponsive to all LSTM layer extensions regarding the integration positions and iteration cycles. The results are quite constant with ± 0.45 % mean accuracy, ± 0.95 % mean precision, and ± 0.6 % mean F_1score (maximum-minimum difference between No. 6 and No. 35 in table 7.3) for the full dataset. For the complex dataset, the results are in a similar value range with ± 1.5 % mean accuracy, ± 0.95 % mean precision, and ± 0.9 % mean F_1score (maximum-minimum difference between No. 10 and No. 23 in table 7.5). However, it is conspicuous that an LSTM layer extension for the deep metadata fusion approach achieves results of the highest accuracy with an integration position after the Conv. 5 layer.

The LSTM layer extension results for the only-vision approach are displayed in tables 7.4 and 7.6. The mean metrics results are also constant, as already seen for the deep metadata fusion approach using the full dataset. However, the only-vision approach has a positive outlier in precision with +0.9 % (No. 26 in table 7.6) in comparison to the approach without LSTM layer extension for the full dataset, cf. table 4.4. Moreover, the LSTM layer extension for the only-vision approach achieves a reversal of the trend that distances between 15 m to 30 m have a higher accuracy than distances under 15 m for the full dataset, see No. 8, 9, 12, 14, and 15 in table 7.6. This is obtained with an LSTM layer integration position after the Conv. 5 in combination with five or seven iteration cycles.

Results of high accuracy in the distance range ($d < 15$ m) are also apparent in the LSTM layer extension for the only-vision approach with the complex dataset, see No. 12 and 21 with 95.5 % accuracy in table 7.4. In general, this also justifies the higher mean accuracy results with the LSTM layer extension for the only-vision approach with the complex dataset, cf. figure 4.11 in subsection 4.7.3.

Tab. 7.3 Detailed experimental results of all LSTM simulations with the deep metadata fusion approach for the complex test dataset.

			Mean Metrics			**Accuracy over Distances d**					
No.	**Integration**	**Iterations**	Accuracy	Precision	F_1score	$d \geq 75$	$75 < d \geq 60$	$60 < d \geq 45$	$45 < d \geq 30$	$30 < d \geq 15$	$d < 15$
1	Conv. 5	3	88.5 %	84.3 %	85.8 %	81.3 %	89.0 %	86.7 %	89.7 %	92.4 %	89.0 %
2	Conv. 5	3	88.7 %	84.5 %	86.0 %	82.0 %	89.1 %	87.3 %	89.1 %	92.4 %	91.6 %
3	Conv. 5	3	89.1 %	85.1 %	86.3 %	83.0 %	89.3 %	88.1 %	89.4 %	92.4 %	90.9 %
4	Conv. 5	3	89.0 %	85.1 %	86.3 %	81.1 %	89.1 %	88.2 %	90.3 %	92.6 %	89.0 %
5	Conv. 5	3	89.1 %	85.5 %	86.5 %	81.7 %	89.4 %	88.0 %	90.6 %	92.3 %	89.6 %
6	Conv. 5	5	89.2 %	85.2 %	86.7 %	82.2 %	89.0 %	88.6 %	90.1 %	92.7 %	89.0 %
7	Conv. 5	5	88.7 %	84.6 %	86.2 %	81.4 %	89.1 %	87.8 %	89.2 %	92.7 %	89.0 %
8	Conv. 5	5	88.8 %	84.4 %	86.1 %	81.1 %	89.3 %	87.2 %	89.7 %	92.6 %	90.9 %
9	Conv. 5	5	88.7 %	84.3 %	86.1 %	81.1 %	89.0 %	87.8 %	89.3 %	92.7 %	90.9 %
10	Conv. 5	5	89.0 %	85.0 %	86.4 %	81.6 %	89.3 %	88.2 %	89.3 %	92.9 %	91.6 %
11	Conv. 5	7	88.6 %	84.5 %	86.1 %	80.2 %	89.1 %	87.8 %	89.2 %	92.8 %	91.6 %
12	Conv. 5	7	88.5 %	84.3 %	85.9 %	81.0 %	88.9 %	88.0 %	88.7 %	92.6 %	90.3 %
13	Conv. 5	7	88.9 %	84.4 %	86.3 %	82.3 %	90.0 %	88.0 %	88.9 %	92.7 %	89.6 %
14	Conv. 5	7	88.8 %	84.7 %	86.2 %	79.3 %	89.7 %	88.3 %	89.8 %	92.3 %	92.2 %

15	Conv. 5	7	88.7 %	84.1 %	86.0 %	81.8 %	89.4 %	88.1 %	89.2 %	91.8 %	90.3 %
16	FC. 1	3	88.8 %	84.1 %	86.2 %	81.8 %	89.4 %	88.2 %	89.6 %	91.8 %	91.6 %
17	FC. 1	3	88.6 %	83.9 %	85.9 %	81.0 %	89.9 %	88.2 %	89.2 %	91.8 %	91.6 %
18	FC. 1	3	88.6 %	83.7 %	85.8 %	81.3 %	89.4 %	87.6 %	89.4 %	91.7 %	91.6 %
19	FC. 1	3	88.7 %	83.4 %	85.9 %	81.1 %	89.6 %	88.0 %	89.2 %	92.0 %	91.6 %
20	FC. 1	3	89.0 %	84.2 %	86.3 %	81.7 %	90.3 %	88.0 %	89.6 %	92.2 %	91.6 %
21	FC. 1	5	88.7 %	84.0 %	86.0 %	82.5 %	90.0 %	87.8 %	89.3 %	91.1 %	91.6 %
22	FC. 1	5	88.8 %	83.8 %	86.0 %	81.0 %	90.2 %	87.8 %	89.4 %	92.0 %	91.6 %
23	FC. 1	5	88.9 %	84.0 %	86.2 %	81.6 %	89.9 %	88.0 %	89.5 %	92.1 %	91.6 %
24	FC. 1	5	88.6 %	83.6 %	85.9 %	81.7 %	89.7 %	88.2 %	89.2 %	91.4 %	91.6 %
25	FC. 1	5	88.9 %	84.1 %	86.2 %	81.8 %	89.9 %	88.5 %	89.5 %	91.5 %	91.6 %
26	FC. 1	7	88.8 %	84.0 %	86.2 %	81.4 %	90.4 %	88.2 %	89.1 %	91.8 %	91.6 %
27	FC. 1	7	88.6 %	83.7 %	85.9 %	81.8 %	89.9 %	87.8 %	88.9 %	91.8 %	91.6 %
28	FC. 1	7	88.6 %	83.9 %	85.9 %	82.2 %	89.7 %	87.8 %	89.0 %	91.3 %	91.6 %
29	FC. 1	7	89.2 %	84.5 %	86.6 %	81.8 %	90.2 %	88.8 %	89.6 %	92.4 %	91.6 %
30	FC. 1	7	88.5 %	83.4 %	85.7 %	81.2 %	88.9 %	87.4 %	89.2 %	91.9 %	92.2 %
31	FC. 2	3	88.6 %	83.6 %	85.9 %	81.4 %	89.3 %	87.6 %	89.2 %	91.9 %	91.6 %
32	FC. 2	3	88.8 %	84.0 %	86.1 %	81.3 %	90.2 %	88.5 %	89.2 %	91.6 %	91.6 %
33	FC. 2	3	88.9 %	84.2 %	86.1 %	81.3 %	89.9 %	88.1 %	89.5 %	92.1 %	91.6 %
34	FC. 2	3	88.9 %	84.0 %	86.2 %	81.2 %	89.6 %	87.9 %	89.7 %	92.4 %	91.6 %
35	FC. 2	3	88.3 %	83.4 %	85.5 %	79.8 %	89.7 %	87.5 %	89.3 %	91.7 %	91.6 %
36	FC. 2	5	88.4 %	83.3 %	85.6 %	80.8 %	89.3 %	87.4 %	89.2 %	91.7 %	91.6 %
37	FC. 2	5	88.8 %	83.9 %	86.1 %	81.3 %	90.2 %	87.9 %	89.4 %	92.0 %	91.6 %
38	FC. 2	5	88.9 %	83.9 %	86.3 %	81.7 %	89.9 %	88.0 %	89.4 %	92.2 %	92.2 %
39	FC. 2	5	88.9 %	84.1 %	86.3 %	81.8 %	90.0 %	88.5 %	89.0 %	92.3 %	91.6 %
40	FC. 2	5	88.8 %	83.7 %	86.0 %	82.3 %	89.7 %	87.8 %	89.4 %	91.7 %	91.6 %
41	FC. 2	7	89.1 %	84.4 %	86.6 %	81.8 %	90.2 %	88.3 %	89.5 %	92.6 %	91.6 %
42	FC. 2	7	88.6 %	83.2 %	85.8 %	81.9 %	89.4 %	87.4 %	88.9 %	92.1 %	90.9 %
43	FC. 2	7	89.0 %	84.1 %	86.3 %	81.4 %	90.4 %	88.2 %	89.3 %	92.4 %	91.6 %
44	FC. 2	7	88.5 %	83.4 %	85.8 %	82.2 %	89.1 %	87.9 %	88.8 %	91.7 %	91.6 %
45	FC. 2	7	88.8 %	84.0 %	86.0 %	82.0 %	89.6 %	87.9 %	88.9 %	92.3 %	92.2 %

Tab. 7.4 Detailed experimental results of all LSTM simulations with the only-vision approach for the complex test dataset.

			Mean Metrics			Accuracy over Distances d					
No.	**Integration**	**Iterations**	Accuracy	Precision	F_1score	$d \geq 75$	$75 < d \geq 60$	$60 < d \geq 45$	$45 < d \geq 30$	$30 < d \geq 15$	$d < 15$
1	Conv. 5	3	85.7 %	82.8 %	82.0 %	71.5 %	82.9 %	86.1 %	89.5 %	89.7 %	89.0 %
2	Conv. 5	3	86.0 %	83.8 %	82.4 %	73.8 %	83.2 %	86.7 %	89.4 %	88.6 %	94.2 %
3	Conv. 5	3	86.1 %	84.2 %	82.6 %	74.3 %	82.6 %	86.3 %	89.2 %	89.6 %	93.5 %
4	Conv. 5	3	86.1 %	83.7 %	82.4 %	73.7 %	83.5 %	86.4 %	89.9 %	89.1 %	89.6 %
5	Conv. 5	3	86.2 %	83.8 %	82.8 %	71.5 %	83.8 %	86.4 %	90.1 %	90.4 %	90.3 %
6	Conv. 5	5	86.4 %	84.4 %	83.2 %	74.7 %	82.9 %	86.9 %	89.6 %	89.5 %	92.9 %
7	Conv. 5	5	86.5 %	84.4 %	83.3 %	75.9 %	84.2 %	86.6 %	89.8 %	89.0 %	92.9 %
8	Conv. 5	5	86.5 %	84.4 %	83.3 %	76.0 %	82.8 %	86.7 %	90.0 %	89.5 %	89.6 %
9	Conv. 5	5	86.2 %	84.1 %	83.0 %	73.7 %	83.1 %	86.6 %	89.9 %	89.1 %	92.9 %
10	Conv. 5	5	86.7 %	85.2 %	83.7 %	76.3 %	84.4 %	87.5 %	89.7 %	89.0 %	92.2 %
11	Conv. 5	7	86.5 %	84.1 %	83.4 %	74.4 %	83.5 %	86.3 %	89.9 %	89.8 %	94.2 %
12	Conv. 5	7	85.7 %	83.1 %	82.5 %	74.5 %	82.9 %	85.6 %	88.5 %	88.9 %	95.5 %
13	Conv. 5	7	86.3 %	84.1 %	83.3 %	74.7 %	82.6 %	86.2 %	89.8 %	89.9 %	92.9 %
14	Conv. 5	7	85.6 %	83.1 %	82.3 %	72.5 %	82.5 %	85.4 %	89.4 %	89.6 %	89.6 %
15	Conv. 5	7	86.5 %	84.2 %	83.4 %	75.8 %	83.6 %	86.7 %	89.6 %	89.0 %	93.5 %
16	FC. 1	3	86.1 %	83.0 %	82.9 %	73.7 %	82.3 %	85.5 %	90.1 %	89.6 %	94.2 %
17	FC. 1	3	86.1 %	83.2 %	82.8 %	74.0 %	83.2 %	85.6 %	90.0 %	89.6 %	89.6 %
18	FC. 1	3	85.8 %	82.8 %	82.5 %	72.9 %	82.8 %	86.0 %	89.6 %	89.3 %	89.6 %
19	FC. 1	3	85.5 %	82.7 %	82.2 %	72.6 %	81.0 %	85.1 %	89.6 %	89.3 %	94.2 %
20	FC. 1	3	86.2 %	83.4 %	83.0 %	74.7 %	83.1 %	86.2 %	89.6 %	89.4 %	94.2 %
21	FC. 1	5	86.1 %	83.3 %	82.8 %	74.0 %	82.5 %	86.2 %	90.0 %	88.9 %	95.5 %
22	FC. 1	5	85.4 %	82.5 %	81.9 %	72.1 %	81.2 %	85.5 %	89.4 %	89.0 %	94.8 %
23	FC. 1	5	85.3 %	82.2 %	81.9 %	72.9 %	82.5 %	85.9 %	89.4 %	87.9 %	89.6 %
24	FC. 1	5	85.8 %	82.8 %	82.6 %	73.9 %	81.8 %	86.0 %	89.8 %	88.7 %	94.2 %
25	FC. 1	5	86.2 %	83.0 %	83.0 %	74.6 %	82.8 %	86.6 %	89.7 %	89.3 %	89.6 %
26	FC. 1	7	85.5 %	82.7 %	82.3 %	72.8 %	82.5 %	86.4 %	89.0 %	88.7 %	89.6 %
27	FC. 1	7	85.9 %	82.9 %	82.7 %	73.3 %	83.5 %	86.3 %	89.4 %	88.9 %	94.2 %
28	FC. 1	7	85.6 %	82.5 %	82.2 %	73.1 %	80.9 %	86.2 %	89.3 %	89.0 %	95.5 %

29	FC. 1	7	86.0 %	83.4 %	82.9 %	74.7 %	83.1 %	86.3 %	89.4 %	88.7 %	94.8 %
30	FC. 1	7	85.7 %	82.5 %	82.5 %	72.9 %	81.5 %	86.0 %	89.8 %	88.7 %	94.8 %
31	FC. 2	3	86.0 %	83.0 %	82.8 %	73.8 %	82.8 %	85.3 %	90.2 %	89.2 %	94.2 %
32	FC. 2	3	85.6 %	82.5 %	82.4 %	72.6 %	82.6 %	84.9 %	89.7 %	89.0 %	94.8 %
33	FC. 2	3	85.6 %	82.4 %	82.4 %	74.0 %	81.9 %	85.7 %	89.2 %	89.1 %	90.3 %
34	FC. 2	3	86.2 %	83.3 %	83.0 %	73.1 %	82.3 %	86.4 %	90.0 %	89.8 %	94.2 %
35	FC. 2	3	86.1 %	83.6 %	83.0 %	74.5 %	82.2 %	86.2 %	90.0 %	88.9 %	94.2 %
36	FC. 2	5	85.8 %	82.8 %	82.5 %	72.7 %	81.9 %	85.9 %	89.8 %	89.6 %	89.6 %
37	FC. 2	5	85.6 %	82.4 %	82.3 %	73.7 %	82.8 %	85.1 %	89.1 %	88.9 %	94.2 %
38	FC. 2	5	86.4 %	83.7 %	83.2 %	74.6 %	82.6 %	86.4 %	90.0 %	89.5 %	96.1 %
39	FC. 2	5	85.6 %	82.5 %	82.4 %	73.1 %	82.2 %	85.8 %	89.4 %	88.7 %	94.2 %
40	FC. 2	5	85.6 %	82.5 %	82.3 %	72.9 %	82.6 %	85.8 %	89.2 %	89.1 %	90.9 %
41	FC. 2	7	86.5 %	83.8 %	83.4 %	73.8 %	83.2 %	87.1 %	90.1 %	89.6 %	94.8 %
42	FC. 2	7	86.2 %	83.3 %	83.0 %	74.0 %	83.1 %	86.7 %	89.6 %	89.0 %	95.5 %
43	FC. 2	7	86.0 %	83.3 %	82.9 %	73.5 %	82.6 %	87.8 %	89.3 %	88.4 %	94.2 %
44	FC. 2	7	85.7 %	82.8 %	82.5 %	73.1 %	81.5 %	86.5 %	89.5 %	88.7 %	94.8 %
45	FC. 2	7	85.9 %	82.9 %	82.7 %	74.4 %	82.8 %	86.1 %	89.5 %	88.7 %	94.2 %

Tab. 7.5 Detailed experimental results of all LSTM simulations with the deep metadata fusion approach for the full test dataset.

			Mean Metrics			**Accuracy over Distances d**					
No.	**Integration**	**Iterations**	Accuracy	Precision	F_1score	$d \geq 75$	$75 < d \geq 60$	$60 < d \geq 45$	$45 < d \geq 30$	$30 < d \geq 15$	$d < 15$
1	Conv. 5	3	93.7 %	93.1 %	93.9 %	88.9 %	91.5 %	93.4 %	95.2 %	96.7 %	95.6 %
2	Conv. 5	3	93.7 %	93.1 %	93.9 %	89.0 %	91.4 %	93.2 %	95.3 %	96.7 %	95.2 %
3	Conv. 5	3	93.5 %	93.0 %	93.8 %	88.6 %	90.9 %	93.7 %	95.0 %	96.6 %	95.6 %
4	Conv. 5	3	93.7 %	93.1 %	93.9 %	89.0 %	91.8 %	93.2 %	95.3 %	96.5 %	95.2 %
5	Conv. 5	3	93.8 %	93.2 %	94.0 %	89.1 %	91.8 %	93.3 %	95.4 %	96.5 %	95.6 %
6	Conv. 5	5	94.3 %	93.6 %	94.5 %	90.4 %	91.8 %	94.3 %	95.7 %	96.7 %	95.6 %
7	Conv. 5	5	93.9 %	93.3 %	94.1 %	89.5 %	92.1 %	93.1 %	95.3 %	96.9 %	95.8 %
8	Conv. 5	5	93.8 %	93.3 %	94.1 %	89.1 %	91.9 %	93.8 %	95.1 %	96.7 %	95.6 %
9	Conv. 5	5	93.7 %	92.9 %	94.0 %	88.5 %	91.3 %	93.8 %	95.6 %	96.7 %	95.2 %

10	Conv. 5	5	93.8 %	93.1 %	94.0 %	88.5 %	91.8 %	93.8 %	95.4 %	96.6 %	95.4 %
11	Conv. 5	7	93.5 %	93.0 %	93.8 %	88.9 %	91.6 %	93.0 %	95.0 %	96.5 %	95.4 %
12	Conv. 5	7	93.9 %	93.3 %	94.1 %	89.8 %	91.5 %	93.8 %	95.0 %	96.8 %	95.8 %
13	Conv. 5	7	94.2 %	93.6 %	94.4 %	90.3 %	91.8 %	93.9 %	95.9 %	96.6 %	95.4 %
14	Conv. 5	7	94.0 %	93.3 %	94.2 %	89.9 %	91.2 %	93.7 %	95.6 %	96.7 %	95.4 %
15	Conv. 5	7	94.0 %	93.5 %	94.3 %	90.3 %	91.8 %	93.7 %	95.3 %	96.7 %	95.2 %
16	FC. 1	3	94.0 %	93.4 %	94.2 %	89.8 %	91.8 %	93.9 %	95.2 %	96.7 %	95.8 %
17	FC. 1	3	93.7 %	93.0 %	93.9 %	88.8 %	91.8 %	93.3 %	95.1 %	96.7 %	95.6 %
18	FC. 1	3	93.6 %	93.0 %	93.9 %	89.1 %	91.2 %	93.2 %	95.0 %	96.8 %	95.6 %
19	FC. 1	3	93.9 %	93.3 %	94.1 %	89.6 %	92.0 %	93.2 %	95.4 %	96.8 %	95.6 %
20	FC. 1	3	93.9 %	93.4 %	94.1 %	89.7 %	91.3 %	93.6 %	95.4 %	96.8 %	95.8 %
21	FC. 1	5	94.1 %	93.5 %	94.4 %	90.1 %	91.8 %	93.7 %	95.6 %	96.8 %	95.8 %
22	FC. 1	5	93.8 %	93.1 %	94.0 %	89.0 %	91.4 %	93.6 %	95.3 %	96.8 %	95.6 %
23	FC. 1	5	94.0 %	93.4 %	94.2 %	89.9 %	92.2 %	93.3 %	95.1 %	96.9 %	95.8 %
24	FC. 1	5	94.0 %	93.4 %	94.2 %	89.8 %	91.6 %	93.8 %	95.4 %	96.8 %	95.8 %
25	FC. 1	5	94.0 %	93.4 %	94.2 %	89.6 %	91.3 %	93.9 %	95.5 %	96.8 %	95.6 %
26	FC. 1	7	94.3 %	93.6 %	94.6 %	90.3 %	92.2 %	94.3 %	95.4 %	97.0 %	96.0 %
27	FC. 1	7	94.3 %	93.6 %	94.6 %	90.3 %	92.0 %	94.3 %	95.7 %	96.9 %	95.8 %
28	FC. 1	7	94.1 %	93.4 %	94.4 %	89.9 %	91.9 %	93.9 %	95.3 %	97.0 %	95.8 %
29	FC. 1	7	94.4 %	93.7 %	94.6 %	90.2 %	92.4 %	94.4 %	95.7 %	97.0 %	96.0 %
30	FC. 1	7	94.2 %	93.6 %	94.5 %	89.7 %	91.9 %	94.5 %	95.5 %	97.0 %	95.8 %
31	FC. 2	3	93.6 %	93.0 %	93.8 %	88.9 %	91.6 %	92.8 %	95.0 %	96.7 %	95.6 %
32	FC. 2	3	93.7 %	93.0 %	93.9 %	88.6 %	91.2 %	93.3 %	95.4 %	96.8 %	95.6 %
33	FC. 2	3	94.0 %	93.4 %	94.3 %	89.5 %	92.1 %	93.9 %	95.4 %	96.9 %	95.6 %
34	FC. 2	3	93.6 %	93.0 %	93.9 %	89.1 %	91.4 %	93.0 %	95.2 %	96.7 %	95.6 %
35	FC. 2	3	93.8 %	93.2 %	94.1 %	89.3 %	91.5 %	93.6 %	95.1 %	96.8 %	95.6 %
36	FC. 2	5	94.0 %	93.3 %	94.2 %	89.3 %	91.8 %	93.8 %	95.4 %	96.9 %	95.6 %
37	FC. 2	5	94.0 %	93.5 %	94.2 %	90.0 %	91.6 %	93.6 %	95.3 %	96.9 %	95.8 %
38	FC. 2	5	94.1 %	93.5 %	94.4 %	90.0 %	92.2 %	93.7 %	95.3 %	97.0 %	96.0 %
39	FC. 2	5	94.0 %	93.4 %	94.2 %	90.0 %	91.8 %	93.2 %	95.4 %	96.9 %	96.0 %
40	FC. 2	5	94.0 %	93.3 %	94.2 %	89.6 %	91.5 %	93.8 %	95.3 %	96.9 %	95.8 %
41	FC. 2	7	94.1 %	93.6 %	94.3 %	89.3 %	92.0 %	94.2 %	95.4 %	96.9 %	95.8 %
42	FC. 2	7	94.2 %	93.5 %	94.4 %	89.6 %	91.8 %	94.3 %	95.7 %	97.0 %	95.6 %
43	FC. 2	7	94.2 %	93.7 %	94.4 %	90.2 %	92.3 %	93.7 %	95.4 %	97.0 %	95.8 %
44	FC. 2	7	93.9 %	93.2 %	94.1 %	88.9 %	91.8 %	93.7 %	95.4 %	96.9 %	95.8 %
45	FC. 2	7	94.4 %	93.8 %	94.7 %	90.9 %	92.6 %	94.0 %	95.5 %	96.9 %	95.8 %

Tab. 7.6 Detailed experimental results of all LSTM simulations with the only-vision approach for the full test dataset.

			Mean Metrics			**Accuracy over Distances d**					
No.	**Integration**	**Iterations**	Accuracy	Precision	F_1score	$d \geq 75$	$75 < d \geq 60$	$60 < d \geq 45$	$45 < d \geq 30$	$30 < d \geq 15$	$d < 15$
1	Conv. 5	3	91.5 %	92.0 %	93.4 %	87.4 %	90.4 %	92.8 %	95.1 %	95.8 %	95.2 %
2	Conv. 5	3	91.6 %	92.5 %	93.6 %	87.5 %	91.5 %	92.9 %	95.2 %	95.9 %	95.8 %
3	Conv. 5	3	91.4 %	92.1 %	93.2 %	86.5 %	91.0 %	92.7 %	94.9 %	95.7 %	95.6 %
4	Conv. 5	3	91.4 %	91.9 %	93.1 %	86.9 %	90.3 %	92.5 %	94.8 %	95.7 %	95.4 %
5	Conv. 5	3	91.5 %	92.2 %	93.5 %	87.3 %	91.1 %	93.0 %	95.0 %	96.2 %	94.3 %
6	Conv. 5	5	91.4 %	92.1 %	93.4 %	86.8 %	91.8 %	93.3 %	94.7 %	95.5 %	95.2 %
7	Conv. 5	5	91.4 %	92.0 %	93.4 %	87.2 %	90.7 %	93.3 %	95.0 %	95.6 %	94.1 %
8	Conv. 5	5	91.4 %	92.0 %	93.2 %	87.3 %	90.9 %	92.2 %	95.0 %	95.4 %	95.6 %
9	Conv. 5	5	91.5 %	92.3 %	93.4 %	87.2 %	91.0 %	93.0 %	94.6 %	95.9 %	96.0 %
10	Conv. 5	5	91.0 %	91.5 %	92.9 %	86.0 %	89.6 %	92.7 %	94.7 %	95.5 %	95.2 %
11	Conv. 5	7	91.3 %	92.0 %	93.2 %	86.8 %	90.9 %	92.3 %	95.0 %	95.5 %	95.4 %
12	Conv. 5	7	91.4 %	92.1 %	93.5 %	87.8 %	90.9 %	92.9 %	94.7 %	95.6 %	95.8 %
13	Conv. 5	7	91.2 %	92.1 %	93.3 %	86.6 %	91.2 %	93.1 %	94.8 %	95.7 %	95.0 %
14	Conv. 5	7	91.5 %	92.3 %	93.5 %	87.8 %	91.1 %	93.7 %	94.8 %	95.3 %	95.6 %
15	Conv. 5	7	91.3 %	92.0 %	93.3 %	87.8 %	90.7 %	92.8 %	94.6 %	95.2 %	95.4 %
16	FC. 1	3	91.4 %	92.0 %	93.2 %	87.9 %	91.2 %	92.9 %	94.2 %	95.6 %	94.3 %
17	FC. 1	3	91.5 %	92.3 %	93.4 %	87.5 %	91.5 %	93.3 %	94.5 %	96.0 %	94.5 %
18	FC. 1	3	91.6 %	92.4 %	93.5 %	88.5 %	91.2 %	93.0 %	94.6 %	96.0 %	94.3 %
19	FC. 1	3	91.5 %	92.4 %	93.4 %	87.7 %	91.4 %	93.2 %	94.4 %	95.9 %	95.2 %
20	FC. 1	3	91.6 %	92.4 %	93.5 %	87.7 %	91.4 %	93.5 %	94.9 %	95.6 %	94.5 %
21	FC. 1	5	91.8 %	92.5 %	93.7 %	88.4 %	91.8 %	93.5 %	94.9 %	95.8 %	94.5 %
22	FC. 1	5	91.6 %	92.6 %	93.8 %	88.1 %	92.1 %	93.2 %	95.1 %	96.0 %	95.2 %
23	FC. 1	5	91.5 %	92.2 %	93.4 %	87.9 %	91.8 %	93.3 %	94.6 %	95.2 %	94.5 %
24	FC. 1	5	91.7 %	92.6 %	93.8 %	88.5 %	91.9 %	93.7 %	95.1 %	95.6 %	94.5 %
25	FC. 1	5	91.4 %	92.0 %	93.3 %	87.7 %	91.9 %	93.1 %	94.5 %	95.3 %	94.3 %
26	FC. 1	7	91.9 %	92.8 %	94.0 %	89.3 %	92.2 %	93.6 %	95.0 %	96.1 %	94.5 %
27	FC. 1	7	91.8 %	92.6 %	93.7 %	89.4 %	91.5 %	93.2 %	95.0 %	95.4 %	94.7 %
28	FC. 1	7	91.7 %	92.6 %	93.9 %	88.4 %	92.1 %	93.5 %	95.3 %	96.0 %	94.5 %

29	FC. 1	7	91.6 %	92.4 %	93.6 %	88.1 %	92.1 %	93.2 %	94.8 %	95.7 %	94.3 %
30	FC. 1	7	91.6 %	92.3 %	93.5 %	88.7 %	91.8 %	93.3 %	94.6 %	95.1 %	94.3 %
31	FC. 2	3	91.4 %	92.2 %	93.3 %	87.4 %	91.2 %	92.6 %	94.7 %	96.0 %	94.5 %
32	FC. 2	3	91.6 %	92.3 %	93.5 %	88.3 %	91.2 %	93.1 %	94.7 %	95.7 %	94.5 %
33	FC. 2	3	91.6 %	92.5 %	93.6 %	88.0 %	91.3 %	93.2 %	94.9 %	96.0 %	95.4 %
34	FC. 2	3	91.5 %	92.4 %	93.5 %	88.1 %	91.2 %	93.1 %	94.8 %	95.9 %	94.1 %
35	FC. 2	3	91.8 %	92.7 %	93.8 %	88.1 %	91.8 %	93.5 %	95.2 %	96.2 %	94.5 %
36	FC. 2	5	91.7 %	92.4 %	93.6 %	88.0 %	91.8 %	93.0 %	95.0 %	96.0 %	94.3 %
37	FC. 2	5	91.5 %	92.3 %	93.5 %	87.7 %	91.5 %	93.0 %	94.6 %	95.9 %	95.2 %
38	FC. 2	5	91.5 %	92.2 %	93.4 %	87.1 %	91.6 %	93.6 %	94.9 %	95.4 %	94.3 %
39	FC. 2	5	91.7 %	92.6 %	93.7 %	88.1 %	91.2 %	93.4 %	94.8 %	96.3 %	95.4 %
40	FC. 2	5	91.5 %	92.0 %	93.3 %	88.6 %	91.5 %	92.9 %	94.5 %	95.0 %	94.3 %
41	FC. 2	7	91.6 %	92.2 %	93.5 %	88.5 %	91.8 %	92.7 %	94.7 %	95.4 %	94.3 %
42	FC. 2	7	91.7 %	92.4 %	93.6 %	88.6 %	92.1 %	93.3 %	94.7 %	95.6 %	94.3 %
43	FC. 2	7	91.4 %	92.1 %	93.3 %	86.7 %	91.5 %	92.6 %	94.9 %	95.9 %	94.3 %
44	FC. 2	7	91.6 %	92.1 %	93.4 %	87.7 %	91.6 %	93.3 %	94.7 %	95.3 %	94.3 %
45	FC. 2	7	91.5 %	92.2 %	93.4 %	87.6 %	92.1 %	92.8 %	94.6 %	95.7 %	94.3 %

7.8 Program Code of the Deep Metadata Fusion Approach

In this section, the program code of the deep metadata fusion approach is explained for the proposed approach configuration.

Listing 7.1 displays the deep metadata fusion approach implementation for convolutional layer two as fusion layer. It uses the proposed CNN topology similar to the AlexNet one. The code is written in python version 2.7 by using the Keras [43] framework version 2.1.2 together with TensorFlow [80] version 1.4.0. Each line is commented in detail. In general, the program code is structured as follows: The data load is briefly specified in lines 21 to line 31. The CNN implementation is contained in lines 38 up to 126. The actual deep fusion code is written from lines 68 to 94. The training and testing of the deep metadata fusion approach is coded from line 131 to the end.

```
1 # import - operation system, keras, and tensorflow frameworks
2 import os
3 import numpy as np
4 import keras
5 import tensorflow as tf
6 from keras.layers.core import Dense, Dropout, Activation, Flatten, Reshape
7 from keras.layers.convolutional import Conv2D, UpSampling2D
8 from keras.layers.pooling import MaxPooling2D, GlobalAveragePooling2D,
     GlobalAveragePooling1D
9 from keras.layers import Input, BatchNormalization
```

```python
from keras.models import Model
from keras.optimizers import SGD

# define - root mean squared error metric function
def pred_rmse(y_true, y_pred):
  return K.sqrt(K.mean((y_pred - y_true) ** 2))

# set - global input parameters
image_size = 256
mfm_size = 64

# read - training images and metadata feature maps and store them in #numpy arrays
#image_train                = [...]
#vector_train               = [...]
#mfm_train_1 to mfm_train_12 = [...]
#mfm_train_default          = [...]

# read - test images and metadata feature maps and store them in numpy #arrays
#image_test                 = [...]
#vector_test                = [...]
#mfm_test_1 to mfm_test_12  = [...]
#mfm_test_default           = [...]

# set - tensorflow session settings
os.environ["CUDA_VISIBLE_DEVICES"]="0"
K.clear_session()
K.set_floatx('float32')

# fusion layer input - metadata feature maps
mfm_input_1  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_2  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_3  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_4  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_5  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_6  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_7  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_8  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_9  = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_10 = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_11 = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')
mfm_input_12 = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')

# fusion layer input - default metdata feature map dflt
mfm_dflt     = Input(shape=(mfm_size, mfm_size,1,), dtype='float32')

# image layer input - images
img_input = Input(shape=(image_size, image_size,3,), dtype='float32')    # input_1

# create - convolutional neural network model
cnn = Conv2D(16, (3, 3), padding='same', activation='relu')(img_input)   # conv_1
cnn = BatchNormalization()(cnn)                                          # batch_1
cnn = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(cnn)     # pool_1

cnn = Conv2D(32, (3, 3), padding='same', activation='relu')(cnn)         # conv_2
cnn = BatchNormalization()(cnn)                                          # batch_2
conv_out = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(cnn)# pool_2

# calculate - adaptive weighting global average
temp_ga  = GlobalAveragePooling2D()(conv_out)                            # shape
    (-,64,64,32)
temp_ga  = Reshape((32, 1))(temp_ga)                                     # shape(-,32,1)
final_ga = GlobalAveragePooling1D()(temp_ga)                             # shape(-,1)

# generate - adaptive weighting global average feature map
temp_ga_fm  = Reshape((1, 1, 1))(final_ga)                               # shape(-,1,1,1)
final_ga_fm = UpSampling2D(size=(mfm_size, mfm_size))(temp_ga_fm)        # shape
    (-,64,64,1)

# weight - multiply each metadata feature map with the global average feature map
aw_mfm_1  = keras.layers.multiply([mfm_input_1, final_ga_fm])
aw_mfm_2  = keras.layers.multiply([mfm_input_2, final_ga_fm])
aw_mfm_3  = keras.layers.multiply([mfm_input_3, final_ga_fm])
aw_mfm_4  = keras.layers.multiply([mfm_input_4, final_ga_fm])
```

```
aw_mfm_5  = keras.layers.multiply([mfm_input_5, final_ga_fm])
aw_mfm_6  = keras.layers.multiply([mfm_input_6, final_ga_fm])
aw_mfm_7  = keras.layers.multiply([mfm_input_7, final_ga_fm])
aw_mfm_8  = keras.layers.multiply([mfm_input_8, final_ga_fm])
aw_mfm_9  = keras.layers.multiply([mfm_input_9, final_ga_fm])
aw_mfm_10 = keras.layers.multiply([mfm_input_10, final_ga_fm])
aw_mfm_11 = keras.layers.multiply([mfm_input_11, final_ga_fm])
aw_mfm_12 = keras.layers.multiply([mfm_input_12, final_ga_fm])

# bulid - metadata feature map tensor by use of the default metedata feature map
mfm_tensor = keras.layers.concatenate([mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt,
     mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt,
     mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, mfm_dflt, aw_mfm_1,
     aw_mfm_2, aw_mfm_3, aw_mfm_4, aw_mfm_5, aw_mfm_6, aw_mfm_7, aw_mfm_8, aw_mfm_9,
     aw_mfm_10, aw_mfm_11, aw_mfm_12]) # shape(-,64,64,32)

# element-wise product - metadata feature map tensor and "conv_out" tensor with identical
      shapes
conv_mfm = keras.layers.multiply([conv_out, mfm_tensor])

# create - convolutional neural network model
cnn = Conv2D(64, (3, 3), padding='same', activation='relu')(conv_mfm)    # conv_3
cnn = BatchNormalization()(cnn)                                          # batch_3
cnn = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(cnn)     # pool_3

cnn = Conv2D(128, (3, 3), padding='same', activation='relu')(cnn)        # conv_4
cnn = BatchNormalization()(cnn)                                          # batch_4
cnn = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(cnn)     # pool_4

cnn = Conv2D(256, (3, 3), padding='same', activation='relu')(cnn)        # conv_5
cnn = BatchNormalization()(cnn)                                          # batch_5
cnn = MaxPooling2D(pool_size=(2, 2), strides=2, padding='same')(cnn)     # pool_5

cnn = Flatten()(cnn)                                                     # flatten_1

cnn = Dropout(0.5)(cnn)                                                  # drop_1
cnn = Dense(1024, activation='relu')(cnn)                                # fc_1

cnn = Dropout(0.5)(cnn)                                                  # drop_2
cnn = Dense(1024, activation='relu')(cnn)                                # fc_2

vec_out = Dense(image_size)(cnn)                                         # output_1

# set - model inputs and model outputs
model = Model(inputs = [img_input, mfm_input_1, mfm_input_2, mfm_input_3, mfm_input_4,
     mfm_input_5, mfm_input_6, mfm_input_7, mfm_input_8, mfm_input_9, mfm_input_10,
     mfm_input_11, mfm_input_12, mfm_dflt], outputs = vec_out)

# set - backward propagation algorithm hyper-parameters
sgd = SGD(lr= 0.00005, decay= 0.0, momentum=0.9, nesterov=True)

# set - model loss function, opimisation algorithm, and metrics
model.compile(loss='mean_squared_error',
          optimizer= sgd,
          metrics=[pred_rmse])

# train - convolutional neural network with its additional hyper-parameters
model.fit([image_train, mfm_train_1, mfm_train_2, mfm_train_3, mfm_train_4, mfm_train_5,
     mfm_train_6, mfm_train_7, mfm_train_8, mfm_train_9, mfm_train_10, mfm_train_11,
     mfm_train_11, mfm_train_default], vector_train, epochs=125, batch_size=200, shuffle=
     True, verbose=2)

# test - trained convolutional neural network model
evaluation_results = model.evaluate([image_test, mfm_test_1, mfm_test_2, mfm_test_3,
     mfm_test_4, mfm_test_5, mfm_test_6, mfm_test_7, mfm_test_8, mfm_test_9, mfm_test_10,
     mfm_test_11, mfm_test_12, mfm_test_default], vector_test, batch_size=200)

# show - evaluation loss and rmse results
print("Evaluation Result: " + str(evaluation_results))

# predict - test output of the trained convolutional neural network model
test_output = model.predict([image_test, mfm_test_1, mfm_test_2, mfm_test_3, mfm_test_4,
     mfm_test_5, mfm_test_6, mfm_test_7, mfm_test_8, mfm_test_9, mfm_test_10, mfm_test_11,
        mfm_test_12, mfm_test_default], batch_size=200, verbose=0, steps=dflte)

# save - predicted test ouptput as *.png image which includes all test sequences
temp_img = Image.fromarray(test_output)
if temp_img.mode != 'RGB':
    temp_img = temp_img.convert('RGB')
save_name = "CNN_Deep_Metadata_Fusion_Layer_2_Test_Output_full_Dataset_v1.png"
temp_img.save(save_name)

# delete - release GPU memory and clear session
del model
K.clear_session()
```

Listing 7.1 Deep metadata fusion approach implementation in python code by means of the Keras and TensorFlow framework with a deep fusion in convolutional fusion layer two and an AlexNet CNN topology.

7.9 High Performance Computing and Processing Time

This work used GPU high performance computing with two NVIDIA GTX 1080 Ti graphic cards. Moreover, the dataset preparation was implemented on a parallel CPU computing cluster with two Intel Xeon E5-2640 v4, each with ten physical cores.

The average processing time per CNN simulation was about two hours per GPU. The total number of simulations (training and testing) was 420: 30 (deep metadata fusion approach investigation of the fusion operator), 60 (deep metadata fusion approach investigation of the metadata effect), 25 (deep metadata fusion approach investigation of the fusion layer), 30 (deep metadata fusion approach configuration of the adaptive weighting), 180 (development of the sequence approach), 10 (only-vision approach), 25 (only-vision approach investigation of the training dataset size) and 60 (only-vision approach investigation of the CNN topology and image size).

The total processing time for the dataset preparations was about 72 hours with all 20 CPU cores for the complete ETL-process[41].

Here, initial CNN simulations, failed CNN simulations due to wrong hyper-parameter setups, and dataset preparations to validate the implementations are not included.

[41]ETL-process: extract, transform, and load data.

Nomenclature

1 Mathematical Abbreviations

Variable	Meaning
$accuracy$	TL2LA accuracy metric
b	bias of a neural network
b	batch index
c	single confusion matrix result ($c \in \{FN, FP, TN, TP\}$)
d	distance to the stop line
e	MFM index
f	frame index
f	focal length
g	global average (scalar)
h_{cam}	world camera height
i	general control variable
$\mathrm{i}_{\mathrm{min}}$	general minimum index
$\mathrm{i}_{\mathrm{max}}$	general maximum index
j	general control variable
k	convolutional feature map index
l	convolutional layer index
l	traffic light column length in the regression vectors or indication vectors, respectively
mv	matching vector
n	iteration or epoch index of a neural network, respectively
ov	output regression vector or output indication vector, respectively
p	tensor height index
p	traffic light column starting position in regression vectors or indication vectors, respectively
$precision$	TL2LA precision metric
q	tensor width index
r	traffic light relevant result ($1 \mapsto$ "relevant" and $0 \mapsto$ "non-relevant")
s	sequence index
t	traffic light index
$\mathrm{t}_{\mathrm{max}}$	traffic light index with the highest traffic light relevant confidence

Variable	**Meaning**
th_{128}	traffic light relevant threshold
tv	target regression vector or target indication vector, respectively
u	camera image height index
u^T	top view image height index
u_H	virtual image horizon
v	camera image width index
v^T	top view image width index
w	camera image depth index
w	weights of an artificial neural network
x	input of an artificial neural network (fully connected layer)
x(v)	lateral world distance in top view image
y	output of an artificial neural network (fully connected layer)
$\hat{y}$	target of an artificial neural network
z(u)	longitudinal world distance in top view image
A	metadata feature map tensor
B	batch size
$\mathrm{B}(p_B)$	Bernoulli distribution
C	traffic light relevant confidence
CI	90 % confidence interval
D	point cloud matrix of disparity map
E_{road}	plane equation of the camera image road part
E_{UV}	UV-plane equation in the camera coordinate system
E_{WU}	WU-plane equation in the camera coordinate system
E	number of MFMs
F	number of frames
FN	false negative assigned traffic lights
FP	false positive assigned traffic lights
F_1score	TL2LA $\mathrm{F_1}$score metric
H	top view transformation matrix
I	recorded (left) camera image
IPM_{config}	fixed IPM configuration parameters
J	top view image
K	number of convolutional feature maps
L	number of lanes in the intersection scenario or sequence, respectively
M	deep metadata fusion tensor

Variable	**Meaning**
M_{alloc}	traffic light to lane allocation matrix
M_{rot}	top view rotation matrix
M_{transl}	top view translation matrix
N	number of neurons in a specific neural network layer
N_D	number of training dataset images
N_E	number of epochs
N_I	number of iterations
N_N	total amount of neurons in a neural network
N_W	total amount of weights in a neural network
O	IPM full panorama image
$O(n)$	O-notation of the complexity performance
P	tensor height
Q	tensor width
R	weight matrix (recurrent) of a convolution neural network
ROI	region of interest of the camera image I
T	number of traffic lights
TN	true negative assigned traffic lights
TP	true positive assigned traffic lights
U	image height
V	image width
W	weights matrix of a convolution neural network
X	input tensor of a convolutional layer
Y	substitution of convolution result tensor
Z	output tensor of a convolutional layer
0_V	zero vector of an LSTM layer
α	flat world assumption angle
β	batch normalisation shift parameter
χ	independent random value of the Bernoulli distribution
ϵ	constant with $0 < \epsilon \ll 1$
γ	ego-vehicle yaw angle
Δ	general comparison variable
η	learn rate of a neural network
κ	kernel size (symmetric)
λ	pooling factor
μ	mean

Variable	**Meaning**
ω_{U^T}	ego-vehicle velocity in top view U^T dimension
ω_{V^T}	ego-vehicle velocity in top view V^T dimension
ϕ	neural network activation function
Ψ	neural network error loss
ρ	pixel density of the imager
σ	standard deviation
σ^2	variance
θ	batch normalisation scale parameter
ξ	substitution of the forward propagation equation

2 Abbreviations

ADAS	Advanced Driver Assistance Systems
ANN	Artificial Neural Network
CCF	Cross-Correlation Function
CI	Confidence Interval
CNN	Convolutional Neural Network
Conv.	Convolutional layer
DAS	Driver Assistance Systems
DF	Deep Fusion
FC.	Fully Connected layer
FN	False Negative
FP	False Positive
GPS	Global Position System
GPU	Graphical Processing Unit
HD-map	High Definition map
IEEE	Institute of Electrical and Electronics Engineers
ILSCRC	ImageNet Large Scale visual Recognition Challenge
IPM	Inverse Perspective Mapping
ITS	Intelligent Transportation System
I2V	Interface to Vehicle communication
LD	Lane Detection
LiDAR	Light Detection And Ranging
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAP	Maximum A Posteriori

Max. Pooling	Maximum Pooling layer
MFF	Multiscale Feature Fusion
MFM	Metadata Feature Map
MLP	Multi-Layer Perceptron
RaDAR	Radio Detection And Ranging
RANSAC	Random Sample Consensus
RA-L	Robotics and Automation Letter
RGB	Red Green Blue
RMSE	Root Mean Square Error
SGDM	Stochastic Gradient Descent with Momentum
SGM	Semi Global Matching
SVM	Support Vector Machine
TLA	Traffic Light Assistance
TLD	Traffic Light Detection
TLR	Traffic Light Recognition
TL2LA	Traffic Light to Lane Assignment
TN	True Negative
TP	True Positive
VGG	Visual Geometry Group

Bibliography

[1] *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*, Jun. 2018. [Online]. Available: https://doi.org/10.4271/J3016_201806.

[2] P. Archambault, M. Delaney, K. Yuzawa, D. Tamberrino, and A. Duval, “Monetizing the rise of autonomous vehicles”, Goldmann Sachs Group, Tech. Rep., Sep. 2015. [Online]. Available: http://pg.jrj.com.cn/acc/Res/CN_RES/INVEST/2015/9/17/f70472c6-f4ad-4942-8eab-3c01f3c717a7.pdf.

[3] H. Seif and X. Hu, “Autonomous driving in the iCity - HD maps as a key challenge of the automotive industry”, *Engineering*, vol. 2, pp. 159–162, Jun. 2016. DOI: 10.1016/J.ENG.2016.02.010.

[4] *Driving innovation - building AI-powered self-driving cars*, NVIDIA Corporation, Jan. 2019. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/.

[5] *Autonomous driving is powerd by Intel*, Intel Corporation, Jan. 2019. [Online]. Available: https://www.intel.com/content/www/us/en/automotive/autonomous-vehicles.html.

[6] H. Zhu, D. Liu, S. Zhang, Y. Zhu, L. Teng, and S. Teng, “Solving the many to many assignment problem by improving the Kuhn–Munkres algorithm with backtracking”, *Theoretical Computer Science*, vol. 618, no. Supplement C, pp. 30–41, 2016, ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2016.01.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397516000037.

[7] T. Langenberg, T. Lüddecke, and F. Wörgötter, “Deep metadata fusion for traffic light to lane assignment”, *IEEE Robotics and Automation Letters*, vol. PP, pp. 973–980, Jan. 2019. DOI: 10.1109/LRA.2019.2893446.

[8] Destatis, “Verkehr: Verkehrsunfälle”, Statistisches Bundesamt (Gesellschaft und Umwelt), Tech. Rep., Dec. 2017. [Online]. Available: https://www.destatis.de/DE/Publikationen/Thematisch/TransportVerkehr/Verkehrsunfaelle/VerkehrsunfaelleJ2080700177004.pdf?__blob=publicationFile.

[9] E. Zinke, “Maßnahmen zu allgemeinen Fahrerlaubnissen”, Kraftfahrt Bundesamt, Tech. Rep., Dec. 2016. [Online]. Available: https://www.kba.de/SharedDocs/Publikationen/DE/Statistik/Kraftfahrer/FE/2016/fe10_2016_xls.html.

[10] NHTSA, "Analysis of fatal motor vehicle traffic crashes and fatalities at intersections, 1997 to 2004", National Center for Statistics and Analysis, Tech. Rep., Feb. 2007.

[11] *Audi expands traffic light information - now available in 10 us cities*, Audi Newsroom, May 2018. [Online]. Available: `https://media.audiusa.com/en-us/releases/241`.

[12] A. Fregin, J. Müller, U. Kressel, and K. Diermayer, "The DriveU traffic light dataset: Introduction and comparison with existing datasets", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3376–3383. DOI: `10.1109/ICRA.2018.8460737`.

[13] R. de Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 333–338. DOI: `10.1109/IROS.2009.5353941`.

[14] J. Ying, J. Tian, and L. Lei, "Traffic light detection based on similar shapes searching for visually impaired person", in *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, Nov. 2015, pp. 376–380. DOI: `10.1109/ICICIP.2015.7388200`.

[15] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1370–1377. DOI: `10.1109/ICRA.2017.7989163`.

[16] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 606–611. DOI: `10.1109/IVS.2017.7995785`.

[17] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 690–700, Mar. 2016, ISSN: 1524-9050. DOI: `10.1109/TITS.2015.2481459`.

[18] J. Müller and K. Dietmayer, "Detecting traffic lights by single shot detection", in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 266–273. DOI: `10.1109/ITSC.2018.8569683`.

[19] A. Fregin and K. Dietmayer, "Learning traffic light colors", in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 709–715. DOI: `10.1109/ITSC.2018.8569746`.

[20] A. Fregin, J. Müller, and K. Dietmayer, "Feature detectors for traffic light recognition", in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 339–346. DOI: 10.1109/ITSC.2017.8317948.

[21] A. Fregin, J. Müller, and K. Dietmayer, "Three ways of using stereo vision for traffic light recognition", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 430–436. DOI: 10.1109/IVS.2017.7995756.

[22] V. John, K. Yoneda, Z. Liu, and S. Mita, "Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching", *IEEE Transactions on Computational Imaging*, vol. 1, no. 3, pp. 159–173, Sep. 2015, ISSN: 2333-9403. DOI: 10.1109/TCI.2015.2480006.

[23] V. John, L. Zheming, and S. Mita, "Robust traffic light and arrow detection using optimal camera parameters and GPS-based priors", in *2016 Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Jul. 2016, pp. 204–208. DOI: 10.1109/ACIRS.2016.7556213.

[24] S. Jung, J. Youn, and S. Sull, "Efficient lane detection based on spatiotemporal images", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 289–295, Jan. 2016, ISSN: 1524-9050. DOI: 10.1109/TITS.2015.2464253.

[25] A. Das, S. Srinivas, and U. Suddamalla, "M2BMT: An automated ground truth generation algorithm for lane detection", in *International Conference on Pattern Recognition Systems (ICPRS-16)*, Apr. 2016, pp. 1–6. DOI: 10.1049/ic.2016.0028.

[26] M. Beyeler, F. Mirus, and A. Verl, "Vision-based robust road lane detection in urban environments", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4920–4925. DOI: 10.1109/ICRA.2014.6907580.

[27] F. Poggenhans, M. Schreiber, and C. Stiller, "A universal approach to detect and classify road surface markings", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1915–1921. DOI: 10.1109/ITSC.2015.310.

[28] D. S. Levine, *Introduction to Neural and Cognitive Modeling*, 2nd. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2000, ISBN: 080582006X.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http:

//papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[30] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *CoRR*, vol. abs/1409.1556, 2014. arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, *CoRR*, vol. abs/1409.4842, 2014. arXiv: 1409.4842. [Online]. Available: http://arxiv.org/abs/1409.4842.

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[33] E. Davies, “Chapter 15 - deep-learning networks”, in *Computer Vision (Fifth Edition)*, E. Davies, Ed., Fifth Edition, Academic Press, 2018, pp. 453–493, ISBN: 978-0-12-809284-2. DOI: https://doi.org/10.1016/B978-0-12-809284-2.00015-0. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128092842000150.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/.

[35] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks”, *CoRR*, vol. abs/1311.2901, 2013. arXiv: 1311.2901. [Online]. Available: http://arxiv.org/abs/1311.2901.

[36] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, G. Ruß, and M. Steinbrecher, “Künstliche neuronale Netze”, in *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Wiesbaden: Vieweg+Teubner Verlag, 2011, pp. 7–11, ISBN: 978-3-8348-8299-8. DOI: 10.1007/978-3-8348-8299-8_2. [Online]. Available: https://doi.org/10.1007/978-3-8348-8299-8_2.

[37] W. Ertel, “Neural networks”, in *Introduction to Artificial Intelligence*. Cham: Springer International Publishing, 2017, pp. 245–287, ISBN: 978-3-319-58487-4. DOI: 10.1007/978-3-319-58487-4_9. [Online]. Available: https://doi.org/10.1007/978-3-319-58487-4_9.

[38] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, “Recent advances in convolutional neural networks”, *Pattern Recognition*, 2017, ISSN: 0031-3203. DOI: https://doi.org/10.1016/

j.patcog.2017.10.013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320317304120.

[39] Q. Liu, N. Zhang, W. Yang, S. Wang, Z. Cui, X. Chen, and L. Chen, “A review of image recognition with deep convolutional neural network”, in *Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part I*, D.-S. Huang, V. Bevilacqua, P. Premaratne, and P. Gupta, Eds. Cham: Springer International Publishing, 2017, pp. 69–80, ISBN: 978-3-319-63309-1. DOI: 10.1007/978-3-319-63309-1_7. [Online]. Available: https://doi.org/10.1007/978-3-319-63309-1_7.

[40] N. K. Manaswi, “Multilayer perceptron”, in *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras*. Berkeley, CA: Apress, 2018, pp. 45–56, ISBN: 978-1-4842-3516-4. DOI: 10.1007/978-1-4842-3516-4_3. [Online]. Available: https://doi.org/10.1007/978-1-4842-3516-4_3.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[42] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *ArXiv e-prints*, Feb. 2015. arXiv: 1502.03167 [cs.LG].

[43] F. Chollet *et al.*, *Keras*, https://github.com/keras-team/keras, 2015.

[44] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[45] P. Neary, “Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning”, in *2018 IEEE International Conference on Cognitive Computing (ICCC)*, Jul. 2018, pp. 73–77. DOI: 10.1109/ICCC.2018.00017.

[46] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network”, in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed., Morgan-Kaufmann, 1990, pp. 396–404. [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

[47] W.-Y. Lee, S.-M. Park, and K.-B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm", *Optik*, vol. 172, pp. 359–367, 2018, ISSN: 0030-4026. DOI: https://doi.org/10.1016/j.ijleo.2018.07.044. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0030402618310167.

[48] Y. Ozaki, M. Yano, and M. Onishi, "Effective hyperparameter optimization using Nelder-Mead method in deep learning", *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 20, Nov. 2017, ISSN: 1882-6695. DOI: 10.1186/s41074-017-0030-7. [Online]. Available: https://doi.org/10.1186/s41074-017-0030-7.

[49] T. Desell, "Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters", in *2017 IEEE 13th International Conference on e-Science (e-Science)*, Oct. 2017, pp. 19–28. DOI: 10.1109/eScience.2017.14.

[50] Y. Wu and K. He, "Group normalization", in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 3–19, ISBN: 978-3-030-01261-8.

[51] C. Wang, H. Yang, and C. Meinel, "Visual-textual late semantic fusion using deep neural network for document categorization", in *Neural Information Processing*, S. Arik, T. Huang, W. K. Lai, and Q. Liu, Eds., Cham: Springer International Publishing, 2015, pp. 662–670, ISBN: 978-3-319-26532-2.

[52] N. Doulamis and A. Doulamis, "Fast and adaptive deep fusion learning for detecting visual objects", in *Computer Vision – ECCV 2012. Workshops and Demonstrations*, A. Fusiello, V. Murino, and R. Cucchiara, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 345–354, ISBN: 978-3-642-33885-4.

[53] T. Dou, L. Zhang, H. Zheng, and W. Zhou, "Local and non-local deep feature fusion for malignancy characterization of hepatocellular carcinoma", in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, and G. Fichtinger, Eds., Cham: Springer International Publishing, 2018, pp. 472–479, ISBN: 978-3-030-00937-3.

[54] A. Guo, B. Yin, J. Zhang, and J. Yao, "Pedestrian detection via multi-scale feature fusion convolutional neural network", in *2017 Chinese Automation Congress (CAC)*, Oct. 2017, pp. 1364–1368. DOI: 10.1109/CAC.2017.8242979.

[55] Q. Chen, X. Meng, W. Li, X. Fu, X. Deng, and J. Wang, "A multi-scale fusion convolutional neural network for face detection", in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2017, pp. 1013–1018. DOI: 10.1109/SMC.2017.8122743.

[56] E. Escobedo Cardenas and G. Camara-Chavez, "Fusion of deep learning descriptors for gesture recognition", in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, M. Mendoza and S. Velastín, Eds., Cham: Springer International Publishing, 2018, pp. 212–219, ISBN: 978-3-319-75193-1.

[57] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 1933–1941. DOI: 10.1109/CVPR.2016.213.

[58] Y. Li, J. Zhang, Y. Cheng, K. Huang, and T. Tan, "Semantics-guided multi-level RGB-D feature fusion for indoor semantic segmentation", in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 1262–1266. DOI: 10.1109/ICIP.2017.8296484.

[59] J. Zhu, J. Zhang, Y. Cao, and Z. Wang, "Image guided depth enhancement via deep fusion and local linear regularizaron", in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 4068–4072. DOI: 10.1109/ICIP.2017.8297047.

[60] P. J. Liang, P. Chondro, J. R. Wu, W. H. Lai, Y. F. Sun, Y. C. Lai, and T. M. Chen, "Deep fusion of heterogeneous sensor modalities for the advancements of ADAS to autonomous vehicles", in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, Apr. 2018, pp. 1–4. DOI: 10.1109/VLSI-DAT.2018.8373245.

[61] M. Oliveira, V. Santos, and A. D. Sappa, "Multimodal inverse perspective mapping", *Information Fusion*, vol. 24, pp. 108–121, 2015, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2014.09.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253514001031.

[62] T. Grosch, "Verfahren zur Transformation von Fahrerassistenzkamerabildsequenzen in ein virtuelles Vogelperspektivenpanorama des Straßenzuges", B.Eng. Thesis, Duale Hochschule Baden-Württemberg, Sep. 2018.

[63] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692. [Online]. Available: http://doi.acm.org/10.1145/358669.358692.

[64] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information", in *2005 IEEE Computer Society Conference on Computer*

Vision and Pattern Recognition (CVPR'05), vol. 2, Jun. 2005, 807–814 vol. 2. DOI: 10.1109/CVPR.2005.56.

[65] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb. 2008, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1166.

[66] T. Langenberg and F. Wörgötter, "Automatic traffic light to ego vehicle lane association at complex intersections", in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 1350–1357. DOI: 10.1109/ITSC.2018.8569421.

[67] I. Iglesias, L. Isasi, M. Larburu, V. Martinez, and B. Molinete, "I2V communication driving assistance system: On-board traffic light assistant", in *2008 IEEE 68th Vehicular Technology Conference*, Sep. 2008, pp. 1–5. DOI: 10.1109/VETECF.2008.437.

[68] B. Bernais, A. Lotz, and H. Pu, "Design and implementation of a traffic light assistance system based on C2X and TSI messages", in *AmE 2016 - Automotive meets Electronics; 7th GMM-Symposium*, Mar. 2016, pp. 1–6.

[69] A. Mazzalai, F. Biral, M. D. Lio, M. Darin, and L. DOrazio, "Automated crossing of intersections controlled by traffic lights", in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 1928–1933. DOI: 10.1109/ITSC.2015.312.

[70] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, Jul. 2016, ISSN: 1524-9050. DOI: 10.1109/TITS.2015.2509509.

[71] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making Bertha see", in *2013 IEEE International Conference on Computer Vision Workshops*, Dec. 2013, pp. 214–221. DOI: 10.1109/ICCVW.2013.36.

[72] N. Fairfield and C. Urmson, "Traffic light mapping and detection", in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 5421–5426. DOI: 10.1109/ICRA.2011.5980164.

[73] X. Li, H. Ma, X. Wang, and X. Zhang, "Traffic light recognition for complex scene with fusion detections", *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 199–208, Jan. 2018, ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2749971.

[74] X. Ding, A. Carlsen, J. Schaefer, M. Marple, D. Klotzbücher, W. Poiger, B. Brust, and F. Trompeter, "Theory and practice: A two-channel automotive RaDAR for three-dimensional object detection", in *2015 European Radar Conference (EuRAD)*, Sep. 2015, pp. 265–268. DOI: 10.1109/EuRAD.2015.7346288.

[75] H. Lee, S. Kim, S. Park, Y. Jeong, H. Lee, and K. Yi, "AVM / LiDAR sensor based lane marking detection method for automated driving on complex urban roads", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1434–1439. DOI: 10.1109/IVS.2017.7995911.

[76] N. Bodla, J. Zheng, H. Xu, J. C. Chen, C. Castillo, and R. Chellappa, "Deep heterogeneous feature fusion for template-based face recognition", in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2017, pp. 586–595. DOI: 10.1109/WACV.2017.71.

[77] R. Zhang, S. A. Candra, K. Vetter, and A. Zakhor, "Sensor fusion for semantic segmentation of urban scenes", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1850–1857. DOI: 10.1109/ICRA.2015.7139439.

[78] L. Qu, S. He, J. Zhang, J. Tian, Y. Tang, and Q. Yang, "RGBD salient object detection via deep fusion", *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2274–2285, May 2017, ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2682981.

[79] X. Zhang, H. Zhang, Y. Zhang, Y. Yang, M. Wang, H. Luan, J. Li, and T. S. Chua, "Deep fusion of multiple semantic cues for complex event recognition", *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1033–1046, Mar. 2016, ISSN: 1057-7149. DOI: 10.1109/TIP.2015.2511585.

[80] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[81] X. Liu, J. Geng, and H. Ling, "Efficient speaker naming via deep audio-face fusion and end-to-end attention model", in *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov. 2017, pp. 405–410. DOI: 10.1109/ACPR.2017.13.

[82] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules", *CoRR*, vol. abs/1710.09829, 2017. arXiv: 1710.09829. [Online]. Available: http://arxiv.org/abs/1710.09829.

[83] H. Weigel, P. Lindner, and G. Wanielik, "Vehicle tracking with lane assignment by camera and LiDAR sensor fusion", in *2009 IEEE Intelligent Vehicles Symposium*, Jun. 2009, pp. 513–520. DOI: 10.1109/IVS.2009.5164331.

[84] R. Altendorfer and S. Wirkert, "Path assignment techniques for vehicle tracking", in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Jun. 2014, pp. 1451–1456. DOI: 10.1109/IVS.2014.6856510.

[85] J. Wang, Z. Liu, S. Yi, and K. Li, "Target vehicle selection based on multi features fusion method", in *2010 IEEE Intelligent Vehicles Symposium*, Jun. 2010, pp. 13–19. DOI: 10.1109/IVS.2010.5548094.

[86] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6526–6534. DOI: 10.1109/CVPR.2017.691.

[87] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi, "Robust camera lidar sensor fusion via deep gated information fusion network", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1620–1625. DOI: 10.1109/IVS.2018.8500711.

Acknowledgement

First and foremost, I would like to thank my doctoral advisor Prof. Dr. Florentin Wörgötter. I am very grateful for all his advice, feedback, and our inspiring discussions. I will remember the cooperation with you in a positive way for a long time.

My thanks also go out to my local advisor Dr. Thomas Hörnlein for his helpfulness and useful hints concerning the design of my experiments, since without it, this dissertation would never reach the level it did.

I would like to express my appreciation to the examining board members Prof. Dr. Carsten Damm, Prof. Dr. Jens Grabowski, Prof. Dr. Stephan Waack, and Prof. Dr. Wolfgang May as well as to Prof. Dr. Minija Tamosiunaite, who was also a member of my thesis advisory committee, for their feedback and their efforts.

Many thanks to my supervisor Dr. Klaus Hermann for his leadership and formal support at Daimler AG.

Furthermore, I would like to thank my colleague Timo Lüddecke for his participation in my IEEE RA-L publication at university and my colleague Andreas Fregin who has shared his recorded dataset with me yet for release at Daimler AG.

Moreover, I would like to express my appreciation to Dr. Simon Tattersal, Dr. Thomas Hörnlein, and Dr. Marcel Langenberg, who have read my publications and my dissertation correction.

I would like to thank all my current and former team members, colleagues and PhD students of other teams, and all working students for the good work environment and support at Daimler AG.

Luckily, there is more to live for than science and research, so many thanks to all my friends with special thanks to my longtime friend Florian Kirchhoff, all of whom have contributed to my social well-being.

I am very grateful to my parents and my brother Marcel for their continuing support and motivation during my research work. Finally, I would like to thank my girlfriend Nicole for her unconditional love.