Deep Learning For Visual Scene Understanding in Autonomous Driving

Ibrahim Halfaoui

Technische Universität München
Lehrstuhl für Datenverarbeitung

Deep Learning For Visual Scene Understanding in Autonomous Driving

Ibrahim Halfaoui

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende(r): Prof. Dr. Sebastian Steinhorst

Prüfer der Dissertation:

1. Prof. Dr. Klaus Diepold
2. Prof. Dr. Björn Menze

Die Dissertation wurde am 20.11.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 25.09.2019 angenommen.

Acknowledgments

Firstly, I would like to thank my direct supervisor Dr. Onay Urfalioglu for his support, guidance and valuable feedback.
Furthermore, I extend my deepest gratitude to my thesis advisor Prof. Klaus Diepold for his extremely valuable counseling, support and understanding of the different challenges I faced during this work. His unique affinity for innovation and efficient problem-solving are true lessons to appreciate and embrace for my future life and career.
Thanks as well to my fellow colleagues at *HUAWEI*, with special mention to Dr. Fahd Bouzaraa for his considerable contribution.
I would like to finish by dearly thanking my family for their love, dedication and support all along my journey.

Abstract

Over the past few years, the emerging concept of *Artificial Intelligence* (AI) has become a prominent buzzword and gained looming popularity as it proved promising in solving complicated tasks. With the start of the AI global age, sensitive human expertise in certain domains might in short no longer be required as machines started to successfully replicate human reasoning patterns and acquired decision-making faculties. *Deep learning* (DL), one of the most trending AI technologies, achieves this by harnessing sophisticated neural networks with complex architectures. Although DL has many application areas, the vehicle industry seems to pay special tribute to this AI tool. To keep up with the rapid evolution, established car constructors as well as ambitious start-up companies have engaged in a fierce competition to fully exhaust the untapped potential of DL. The ultimate goal is to conceive an autonomous system able to make reasonable decisions in challenging driving situations. For this, solid understanding of the environment and proper sensory data interpretation from cameras in particular are key aspects.

In this context, we provide a solution for optimizing scene understanding for autonomous agents by leveraging in-house designed Convolutional Neural Network (CNN) architectures and customized training procedures. We propose a robust CNN model for class-level semantic segmentation with instance-level distinction of important traffic components such as cars. Additionally, we explore the advantage of combining segmentation information with further computer vision tasks necessary to improve visual scene understanding, notably depth/disparity estimation. The conducted exploration revolves around the examination of multi-tasking models that use cooperative optimization to jointly learn different tasks simultaneously. Aside from the analysis of the performance improvement induced by incorporating segmentation information, this work also proposes an alternative way to perceive and design training procedures for such models. In fact, we present a new method for training multi-tasking CNNs conceived as a competitive game that boosts performance, guarantees design flexibility and overcomes data-related limitations.

Contents

List of Figures

1 Introduction

1.1 Autonomous Driving

The safety of users has always been the driving force guiding the automotive industry since its inception towards more advancement and innovation. However, the average statistics of yearly deaths and injuries in car accidents worldwide keeps going upwards with 1.3 million deaths and between 20 and 50 million injuries [7]. Since human errors are statistically the major cause of these accidents (more than 90% of the accidents are caused by human failure [8]), the notion of autonomous and aware vehicles started to rapidly gain interest over the past few years. The community sets big hopes on the fact that this technology can contribute to lowering these numbers by eliminating the main causing factor: human failure.
Although this concept has been around for a while (e.g. the "American Wonder" project [9] of the Chandler Motor Car Company in the 1920s), developing fully safe and reliable self-driving vehicles was, up to recently, considered as a science-fiction fantasy. It has been doubtfully questioned until the emerging AI revolution turned the tables and pushed it back to the surface as a serious scientific research topic.
Big Tech companies (Huawei, Google, Waymo, Apple, Tesla, Nvidia, Uber, Lyft, etc...), as pioneers of the AI era, are paying special attention to the topic of driverless cars. They intend through their disruptive products, e.g. the Autonomous Google Car, to guarantee safer (fewer accidents) and more convenient (less traffic jams) driving experience for the users. Forecasting the emerging market change, established car manufacturers started as well to race rashly in this direction despite the numerous uncertainties and challenges in the way towards full autonomy for future vehicles.
Regarding the new market trends, a normalized lexicon shared among field actors was needed to bring more clarity and structure to the topic. Hence, the Society of Automotive Engineers (SAE) released in 2014 the international standard *J3016* defining in details the various autonomy levels for vehicles. The updated version from 2016 is summarized in Fig. 1.1. Based on these standards, autonomous systems can be classified into six different levels regarding their sophistication.

- Level 0 - No Automation: The human driver is fully controlling the vehicle at all times.
- Level 1 - Driver Assistance: The driver is still in control and bares full responsibility in monitoring the environment and performing the driving functions. However, few functions are undertaken by the vehicle such as adaptive cruise control.

- Level 2 - Partial Automation: Although the driver remains engaged with the driving, multiple primary functions could be undertaken by the vehicle.
- Level 3 - Conditional Automation: All aspects of driving can be fully undertaken by the vehicle under certain conditions. The driver is still required and expected to be ready to intervene upon notice.
- Level 4 - High Automation: The vehicle is able to perform all driving functions without human intervention but only in certain use cases. The driver can take control any time he wants.
- Level 5 - Full Automation: The vehicle is able to perform all driving functions under any conditions without human intervention.

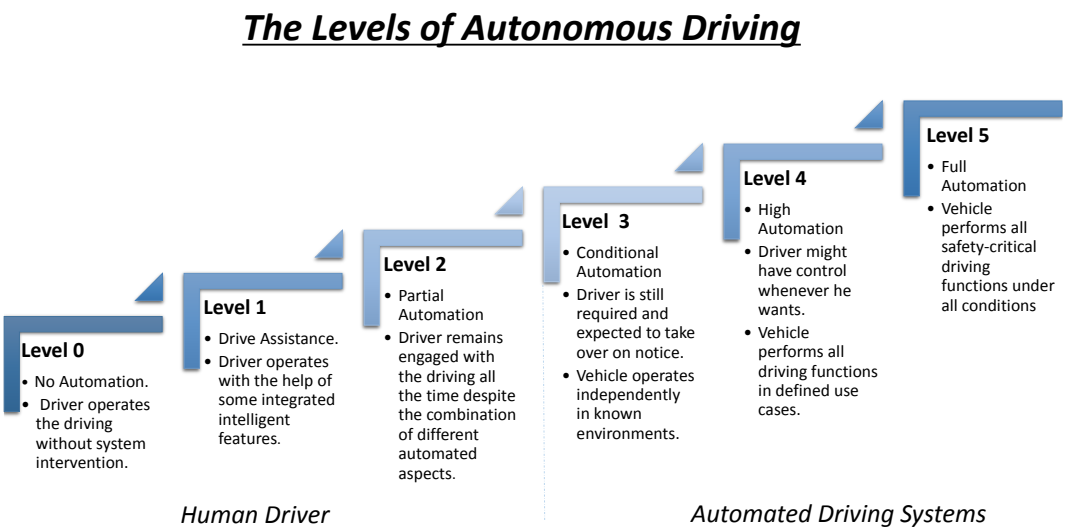


Figure 1.1: The levels of autonomous driving.

Starting from Level 3, the vehicle autonomy becomes very important as the complexity of its expected functions increase significantly compared to previous levels where the human driver undertakes most driving functions. However, the biggest leap happens between levels 3 and 4 where the responsibility for monitoring the environment completely shifts to the system. To our knowledge, there are no Level 4 production vehicles available to consumers yet. Finally, the ultimate goal aimed by the community is level 5 as it is the level where no provisions for human control are expected under any conditions. In theory, the user is able to enjoy the ride without concern about the act of driving.
To reach this level, a complex chain of tasks should be reliably performed by the system in

order to negotiate any driving situation. The process can be divided into three important steps as shown in Fig. 1.2. Each step encompasses various tasks expected to be perfectly realized by the system. Any small failure could have huge effects on the final behavior of the vehicle in traffic. Sensing and decision steps could be considered as classical topics that have been examined for a while. However, the perception part is the one part where classical methods proved limited and the emerging AI methods are likely to deliver the most. This is the step where more "intelligence" for current systems is required. The concept of perception encompasses the faculty of properly analyzing the incoming sensory information, using it to generate understanding of the environment and perceiving the traffic as a human does.

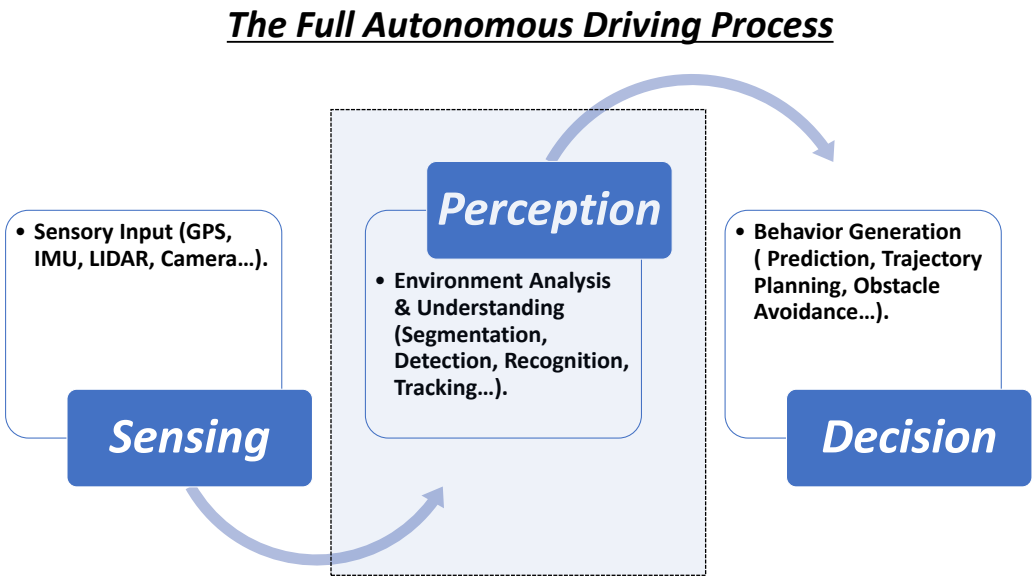


Figure 1.2: The flowchart of the full autonomous driving process (Level 5).

In reality, the enthusiasm about improving perception started high following the emersion of robust DL-based technology but suffered unexpectedly a huge blow after the tragic death of Joshua Brown in a car accident in May 2016. Accusations were promptly addressed to his Tesla's autopilot upon its failure to identify the trailer of a white truck that was blocking the path. In other words, the perception has failed. Obviously, pessimist voices started immediately claiming that the community is deliberately over-pledging and under-delivering on the advantages of the maturity-lacking methods. The most optimists, however, continue to believe that a long transition process needs proper time and enough experimentation in order to fully optimize the underlying technology.

1.2 Scene Understanding In Autonomous Driving Scenarios

1.2.1 Motivation & Background

After the tragic accident of Joshua Brown in 2016, not only Tesla (the manufacturer of the car involved in the accident) but also authorities and other field actors working on the subject of driverless vehicles started to reconsider certain aspects of the perception problem and seek alternative ways to approach it. Thereupon, focus has been shifted towards acquiring more data on the topic and gathering as much reliable content as possible from different autonomous driving experiences. Along with gaining an insightful overview on user-vehicle interaction in critical situations, this data pool is essential for analyzing the human perception faculty and trying to learn from it as much as possible. In fact, DL applications require reliable knowledge sources and large data pools like this to guarantee proper learning. The concept behind this technology boils down to the fact of using large artificial neural networks to solve high complexity tasks. A network that is built as a layered architecture of different blocks simulating computational entities is indeed a mathematical abstraction of inter-connected brain neurons. Through training, this ensemble is taught to approximate mapping functions between pre-disposed input data and ground-truth labels. In reality, this procedure used to be quite challenging. However, it is fortunately no longer the case thanks to the significant culmination reached on hardware (computational power capacities) and software levels (robust, effective and multi-usable neural network architectures).
In the context of autonomous driving, the ultimate goal is to properly teach machines to equal or/and surpass humans in the decision making process using DL. However, a worthy replacement let alone surpassing of humans requires that the autonomous vehicle must have a fully reliable perception of its navigation environment. If not, it will not manage to take proper driving decisions and might cause accidents.
In the mentioned example, investigations of the 2016 Tesla accident proved indeed that a wrong perception of the traffic scene might be the main cause of the crash. It was asserted that the autopilot failed to identify the white side of a driving truck against the brightly lit sky, otherwise it would have released the brakes before hitting its trailer and passing underneath it. Surprisingly, the same autopilot did not have any issues perceiving the environment all along the thirty-seven-minute journey before the crash. It even managed to avoid a second crash after hitting the truck. After the car slid off the way due to the collision, brakes were deployed to avoid hitting a tree on the sidewalk. According to the recorded car data and the human testimonies of few witnesses on the spot, it was very unlikely that Mr Brown released the brakes since he should have already passed away during that time. The conclusion is that the autopilot recognized the tree, applied the brakes and even released the airbags before hitting it. Obviously, the system was able to properly understand the presence of a tree in the second situation but not the track in the first one, that is why the decision came late at a high cost.
As a synthesis, Mr. brown's unfortunate death comes as a closing statement to rest the

case in favor of enforcing environment understanding for autonomous systems under all possible conditions. Even the finest details in the scene shall be well perceived for such systems in order to avoid further mis-happenings in the future. Although several sensory inputs such as (GPS, IMU, LIDAR, etc..) could be made available to an autonomous system, visual recorded data (image/video) comes at the pinnacle of the directions to investigate especially after analyzing the causes of the aforementioned accident. Relying on proper visual information helps depict/recognize objects, estimate their real-world distances and produce real-time 3D maps. Having this, the system can properly decide when to stop, start and react. Therefore, in-depth consideration to the visual data, as the major incoming information carrier, becomes necessary in the goal to offer the system better understanding of its environment.
What kind of cues shall be leveraged and how to get best understanding of real traffic scenes from visual data?

1.2.2 Visual Scene Understanding

Many species are blessed with remarkable visual systems that perform astonishingly fast and good in understanding the complex 3D real-world. Humans as an example [10] are able to understand a complex world scene just from its projection on the retina by locating, characterizing and recognizing objects and features at a single glance. This understanding process consists of successive tasks (high level scene classification, annotation, segmentation...) performed sequentially to form a coherent framework called "visual understanding process". Once applied, this process delivers an output where the hierarchy of the semantic information inherent in the projected image is encoded. By decoding the resulting information, a proper comprehension of the scene content can be acquired.
In the context of computer vision, we can simulate this "visual understanding process". Its final output is generally a compact representation of the information carried by an image represented under a simplified form. In general, this representation is easier to interpret and consists of a set of well-separated components that share common properties. These properties could be defined at different levels of abstraction. The way they are defined is determinant to how the scene is perceived. In fact, parsing the image into coherent parts without taking into consideration the semantic meaning might turn out a confusing factor for interpretation. Hence, the segmentation strategy required should necessarily deliver semantically meaningful partitions. That is why, it is referred to as "semantic segmentation".
Regardless the required level of abstraction, the task of semantically segmenting an image is still a complex task. It is also an important step of the simulated understanding process. This step is responsible for moving from inferring the coarse layout (high-level features) of a scene to retaining its fine details (intermediate and low-level features). Although "understanding" a scene might seem a vague concept conditioned by subjectivity, it is tightly related to the quantity (level of details depicted) and quality (accuracy) of the semantic segmentation's outcome.

The importance of semantic segmentation for the understanding process explains the fact that it is still one of the most addressed problems in computer vision. It has, however, reached a kind of saturation level before DL emerged. Despite their popularity, traditional algorithmic techniques for semantic segmentation are leaving the lights gradually in favor of DL-based methods. The latter indeed have already gained a considerable lead against the former ones in various computer vision related tasks, semantic segmentation in particular.
Nevertheless, the previously mentioned accident proves that maturity stage is far from reached for the new technology. For proof, the on-board autopilot failed to differentiate between the brightly lit sky and the white side of the truck in a critical situation. As rare as it might be, this error resulted in a "mis-understanding" and a confusion in the analysis of the scene components that caused consequently the "mis-judgment/wrong decision" taken by the system at that instant. Regrettably, this error came at a very high cost to prove that state-of-the art performance of DL-based semantic segmentation still needs optimization. Our presented research work tries to bring a small contribution in the goal to tackle the issue from a specific point of view and under well-defined requirements. The suggested approach could not in any way be an ultimate solution to the semantic segmentation problem but rather an alternative to cope with specific aspects of the issue. It is the author's suggestion as an answer to the question within a specific business context and with clear pre-defined project requirements.

1.3 Project Specifications

The final business goal behind the project, within which this research work has been conducted, is to produce a software solution for driverless vehicles to realize safe navigation in different traffic scenarios without potential human interference. To achieve this, improving the understanding of traffic scenes for autonomous driving systems is required.
For this project, cameras are supposed to be the primary controller. That is why, the scope of the presented work will not cover further sensory information like (GPS, IMU, Radar/LIDAR, etc...). We will restrict ourselves to visual scene understanding and the potential ways to improve it through semantic segmentation as well as other available cues.
During the project planning, several pre-defined requirements were set up to guarantee a fair evaluation in the final phase and to help restrict the scope of the explored terrain. Requirements considering the hardware dictate that the used visual system has to be multi-view at least stereo. On Software level, the required solution has to be based on DL and convolutional neural networks (CNNs). All available visual information offered by the hardware can be considered and processed accordingly without restrictions. This includes many processing possibilities e.g. image segmentation, edge detection, disparity/depth...
Among all traffic participants, cars present in the environment of the self-driving vehicle are of much importance. They are the main traffic agent to mind during the navigation together with pedestrians. They also generally occupy a considerable part of images depicting driv-

ing scenes especially within cities. They are frequently present in the camera field of view and have a huge impact on the inherent visual information (e.g. occlusions). In fact, they are statistically the most present traffic participant in the majority of self-driving datasets. As an example, the statistical analysis of the most popular ones such as KITTI [11] and Cityscapes [1] shows that the car class among all others has the biggest absolute average number of labeled instances with $41 \times 10e^3$ instances in Cityscapes and $30.03 \times 10e^3$ in KITTI. Cars present in an urban scene could be stationary or moving at the time of image capture. This state is an important aspect to consider for the autonomous system during the perception stage and the decision making step that follows. For this, it needs a proper distinction between dynamic and static vehicles. The anticipation of the traffic behavior varies a lot depending on the state of other present vehicles in the scene. As an example, considering only filtered feature points lying over segments of stationary cars, could hugely lower the error rate during the matching step of the simultaneous localization and mapping procedure such as ORB-SLAM2 [12]. This consequently gives the vehicle a better understanding of its dynamic surrounding environment. In general, most of the object classes potentially present in an urban scene are easy to categorize into dynamic or static elements. For example, a building, a road, a tree are obviously static objects. However, it is hard to extrapolate this over cars since they could be in motion, parked or just not moving with the engine on at the time of capture. Therefore, separating instances (segments corresponding to independent single cars) within the car class might be of higher importance for the current project phase in comparison to the other classes. For the sake of simplicity, we focus in this work on the instance semantic segmentation of just cars as a specific example. Further extension over other classes might be considered for later project stages.

1.4 Goals and Structure of the Thesis

Considering the project specifications detailed in the previous section, the main goals of this work could be defined as follows:

- Develop a CNN-based solution for urban scene understanding suitable for autonomous driving applications. This implies that we seek to provide an autonomous vehicle navigating an unfamiliar environment with the faculty to perceive the traffic, analyze it and interpret it in a proper way (as close as possible to humans). The goal is to use this knowledge to optimize the driving decisions taken by the system during the journey.
- As the concept of scene understanding is quite vague. We restrict ourselves to focus on the pre-defined project specifications. The proposed solution shall be limited by the available software and hardware setup previously discussed.
- Considering our pre-fixed working range, we need to start by defining the different levels of understanding required by the autonomous system. As a first step, a robust

model for class-level image semantic segmentation should be proposed. This solution is then to be extended with an instance-level distinction between the car entities. The resulting model should be able to segment a captured traffic scene into different object classes and to distinguish independent instances within the car object class at the same time.

- As a next step, we seek to enrich our visual scene understanding solution by incorporating distance information about the different components of the scene at hand. Disparity estimation must be examined separately as well as in cooperation with semantic segmentation. A thorough performance examination shall be realized and a comparison between single and multi-tasking frameworks is to be done.

- Certain real-case challenges need as well to be tackled. For instance, we need to take into consideration the difficulty to have real ground-truth data available to train supervised models to perform semantic segmentation and disparity estimation. Moreover, we need to guarantee consistent robustness (generalization performance) and high accuracy of the solution under different conditions such as getting maximum profit of the data at hand and overcoming the inflexible implementation constraints likely to affect the performance of multi-tasking neural networks.

Considering our previous goals, we start first by proposing a CNN-based approach for image semantic segmentation in Chapter 2. The solution is then extended from class to instance level segmentation for cars in particular. A final combined model is then presented and examined upon its ability to perform robust classification of different components in a traffic scene, to generalize well to various conditions and to distinguish between independent car instances. In Chapter 3, we further enrich the understanding capability of our proposed model by extending it with additional tracks. Different applications are considered such as disparity plane estimation (learning disparity on segment level) and supervised disparity estimation (learning disparity on pixel level). Both applications are combined with semantic segmentation within classical multi-tasking frameworks and their performance is examined to verify the advantage of simultaneously learning related tasks. In Chapter 4, we first propose an alternative solution to perform pixel-wise disparity estimation in a self-supervised way. Then, we study the effect of learning semantic segmentation in parallel to tasks of such unsupervised nature. Besides, a new strategy for training multi-tasking models is presented to overcome the limiting constraints of data availability and to improve upon classical state-of-the-art performance. In Chapter 5, we summarize our conclusions and discuss the deliveries of the realized research work as well as its limitations.

2 Segmentation for Autonomous Driving

2.1 Introduction

Segmentation is a broad concept and a critical task in image processing. We can define it as the partitioning of an image into distinct homogeneous set of pixels having the same attributes and sharing the same properties/characteristics such as colour, texture or intensity, etc...
The expected result is always a simplified representation of pixel-level encoded information under the form of independent uniform segments that can be separately manipulated.
The importance of image segmentation, one of the most explored problems in the computer vision field, derives from the need for simplified data representations that carry as much global information as possible. The representation of visual information as pixel matrices (texture images) is of high complexity and does not generally allow for holistic understanding of the depicted scenes. Therefore, extensive effort has been allocated to simplify it under the form of segmented sets.
Depending on the pre-defined criteria, the strategy for partitioning an image into different pixel sets might vary. At the end, the outcome always relates tightly to the properties considered for grouping the pixels. This could result in segments that are not semantically meaningful but present certain homogeneity. The segmentation is then called semantically non-aware. However, if the segments correspond to objects, parts or groups of objects then the outcome is semantically-aware.
For semantically-aware methods, the output also depends on the level of granularity required/needed. Once the required detail-level is reached (class, object, part of object), no further partitioning of the image will be necessary. For example, distinguishing different object-classes (trees, cars, pedestrians, sky ,road, etc...) and identifying instances corresponding to different objects belonging to the same object-class (separate each car entity, distinguish every single tree, etc...) are both semantic segmentation tasks that rather need different levels of object abstractions. In the first case, pixels belonging to any affiliated objects of the same object-class are similarly labeled without need of further detailing. This is called *class-level semantic segmentation*. In the second, obvious distinction between objects within the same class is additionally required. The task is referred to as *instance-level semantic segmentation*.
Regardless its awareness standard, semantic segmentation remains always a useful task that has many standard application areas such as security, military, medicine, astronomy, physics, industry, etc... It could be particularly important for autonomous driving scenarios where a high segmentation quality will help the system better perceive the world.

2.2 State-of-the-Art Survey

2.2.1 Non-Semantic Segmentation

Partitioning an image without taking into consideration the semantic meaning is a basic task. It only depends on the pre-defined separation criteria. Considering these, we can distinguish different families. First, we note the simple non-contextual thresholding segmentation where pixels are classified into different categories considering their carried intensities. Separation is done with respect to a threshold that might be dynamically, statically, intensity-based, color-based or entropy-based [13] set. Second, there are the contextual segmentation methods. For these, pixels are grouped considering certain image features such as similarity or spatial closeness [14]. Moreover, various further non-semantic methods can be mentioned: Super-pixel segmentation [15], graph-based segmentation [16], quick-shift segmentation [17] and watershed segmentation [18].

2.2.2 Semantic Segmentation

Unlike non-semantic segmentation, the semantic one is a hard-shell classification of pixels into a set of segments with pre-defined labels. These correspond always to pre-known fixed classes (see Fig. 2.1). Depending on the required detail-level, semantic segmentation can be either class or instance-aware. Considering the abundant number of methods dealing with the problem, we restrict ourselves to just exploring DL-based methods targeting autonomous driving applications. That is why, particular focus on datasets for such applications (e.g. Cityscapes, KITTI, etc...) will be given.

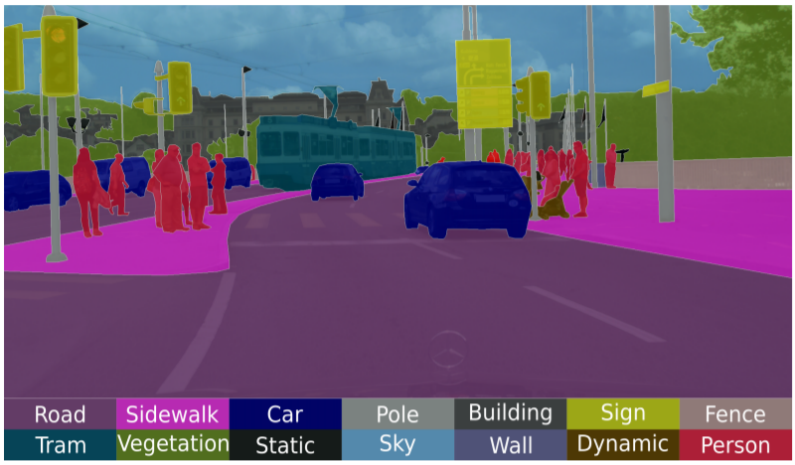


Figure 2.1: Semantic segmentation of a scene from the Cityscapes dataset by Cordts *et al.* (2016) [1] recorded in Zurich (courtesy of Janai *et al.* [2])

Class-level Semantic Segmentation

For Semantic segmentation, Convolutional Neural Networks (CNNs) encountered a considerable incremental success over the past few years. In [19], a review of the current most performing approaches was realized. One of the earliest CNN approaches aiming to perform semantic segmentation is the patch-based classification model introduced by Ciresan *et al.* in [20]. The used architecture was based on fully-connected layers as output classifiers (vector of likelihoods with a value of each class) for each pixel with the additional consideration of its surrounding neighborhood (squared region centered on it). The disadvantage of this architecture is that it requires a fixed-size input image. It processes it into a succession of extracted patches that are independently handled in every step. Consequently, much of the inherent pixel correlation get lost.
In 2014, Long *et al.* introduced a new type of end-to-end segmentation models in [21]. The new architecture called Fully Convolutional Networks for Semantic Segmentation (FCN) represented a turning point that marked the impact of CNNs. They become able to solve many high-complexity tasks, not only segmentation, in a much efficient end-to-end way. With FCN started the trend of encoder-decoder architectures for pixel-to-pixel learning. Unlike sliding window architectures, this new type has an encoder part with convolutional layers where different cues from the input image are learned and encoded into feature maps, and a decoder part with transposed convolutional layers (layers where a backward/-transposed 2D convolution is applied to up-sample the input into more dense output using learned kernels, this operation is widely known as deconvolution) where the outcoming feature maps are back projected into the pixel-space. With this layout, pixel-to-pixel estimates for the incoming input images could be recovered in much shorter runtimes.
Adopting the same archetype, various encoding-decoding architectures have been suggested. They all share the same core but slightly differ on the design-level such as UNet by Ronneberger *et al.* [22], SegNet by Badrinarayanan *et al.* [23], DenseNet by Gao *et al.* [24], Enet by Pazke *et al.* [25], LinkNet by Chaurasia *et al.* [26], PspNet by Zhao *et al.* [27], RefineNet by Lin *et al.* [28], G-FRNet by Islam *et al.* [29], Decoupled-Net by Hong *et al.* [30], Deeplab with by Chen [31] [32] [33], ResNet by he *et al.* [34].

All these architectures share the two-stage design with encoding and decoding parts shown in Fig. 2.2. Nevertheless, each single model comes with specific features that characterize its performance. UNet [22] for example replaces the pooling layers by transposed convolution operators with large channel numbers in the target to optimize localization for high resolution. After eliminating the fully connected layers, the resulting ladder structure embraces a "U" shape. It also has few skip-connection between the encoding and decoding parts used to minimize the detail loss after the extensive dimension shrinkage. SegNet [23] proposes a quite similar design with an additional transfer of max-pooling indices from encoder to decoder. This new property aims to particularly improve non-linear up-sampling. Since the up-sampled feature maps are usually sparse, a convolution with trainable filters is performed to end up with more dense outputs.
The naming DenseNet [24] by Gao *et al.* does not refer to the output's density but rather

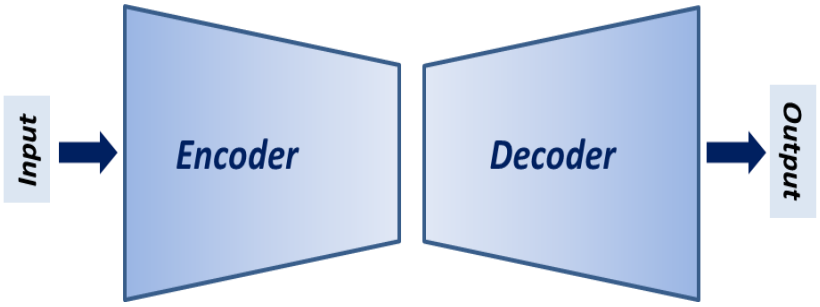


Figure 2.2: The encoder-decoder network layout.

to the concept of "feature reuse". This consists of using consolidated inter-links between different layers to enforce replication of feature maps. Applying this on different levels of the deep architecture improves the learning and guarantees the reduction of processing time and computation load during the training phase. Hence, the resulting architecture is simpler and more efficient in comparison to heavy models such as ResNet [34]. The latter is a deep architecture introduced in 2016. It is not designed as a pile of stacked layers connected to each other to form an acyclic graph but uses additional extra connections to surmount the problem of vanishing gradients. In fact, the problem of saturation caused by weak gradients at the top of the architecture is a main issue for training deep networks. That is why, ResNet leveraged identity shortcut connections based on skipping layers to avoid it. The underlying assumption that residual loss functions are easier to optimize was inspired by Highway networks where the information flow into shortcut connections is controlled by entering gates. This idea proved efficient in reducing saturation problems, making ResNet an important milestone in the development of CNN-based solutions.
Moreover, the recently released PspNet [27] proposes a design inspired by the ResNet. It uses dilated convolutions and pyramid spatial pooling modules (a set of large kernel pooling layers) to improve the aggregation of global context information necessary to approximate the distribution of segmentation classes over large image regions. Unlike ResNet, it applies "intermediate supervision" that revolves around interposing auxiliary loss functions at intermediate layers to improve back-propagation.
Just like PspNet, the different Deeplab versions [31] [32] [33], make use of dilated convolutions implemented in cascade or in parallel to form an efficient Atrous Spatial Pyramid Pooling (ASPP) module. This module is able to capture background context-based information at different scales/resolutions. The authors suggested various improvements for each released version. In the last one, they eliminated Conditional Random Field (CRF)

modules and replaced them with highly optimized ASPPs combined with batch normalization.
ENet [25] and LinkNet [26] are parameter effective versions of the popular ResNet. They are more suitable to real-time implementations, mobile applications and embedded platforms. They propose similar ladder architectures with limited number of parameters and reasonable performance compared to state-of-the-art. Both models focus on offering the required tradeoff between fast inference and high accuracy. In fact, they deploy a well-studied channel reduction scheme that decreases considerably the cardinality without affecting the performance.
Other models were introduced for optimizing the refinement of the segmentation output such as RefineNet [28] and G-FRNet [29]. Both are able to produce densely labeled segmentation for high-resolution input by effectively integrating contextual information. For this, RefineNet uses a multi-path structure with long-range residual connections. It processes data in multiple resolutions for each stage and then forwards the results to the next one without loosing background context-based features. Unlike PspNet, RefineNet is much less memory consuming and does not require dilated convolutions thanks to the residual blocks fused by identity mapping. Similarly, G-FRNet is also effective in end-to-end labeling tasks for high-resolution inputs. It specifically addresses the problem of poor quality caused by missing forward(-passed) information. It achieves this by interposing gated feedback units in order to intercept this information and modulate its flow from encoder to decoder. By only allowing the information necessary for segmentation, ambiguities become easier to filter out which consequently improves the refinement quality. In [30], a semi-supervised network for segmentation was introduced for the first time. The model called Decoupled-Net does not approach the problem as a single task but decouples the classification from the segmentation track. Each network branch is independently responsible of its own task. They just communicate through a bridging layer where information is passed. This flexible design makes it easier to employ separate pre-trained classification networks for all available class regions previously estimated by the segmentation track and communicated over the bridging layer.

Instance-level Semantic Segmentation

Instance-level semantic segmentation is a challenging classification problem, especially in the context of autonomous driving. Unlike object detection that addresses the problem of estimating bounding boxes encapsulating the objects of interest, instance segmentation tends to exactly isolate independent objects. It consists of applying pixelwise labeling for each independent entity (local segmentation) also within the same class to allow for proper differentiation between objects (even if they belong to this same class).
Obviously, various methods tried to approach the problem as a two-stage task. First, a region of interest (RoI) for every object is estimated to limit the processing area. Then, a local segmentation procedure is applied within the RoI to extract and segment the object of interest. Most of these methods such as [35] and [36] are based on the popular

Region-based Convolutional Neural Network (R-CNN) [37] proposed by Girshick *et al.* for estimating object RoIs as a first step. For the second stage, He *et al.* presented in [35] a designed layer for properly aligning RoIs with extracted features. This layer produces no misalignment artifacts since it does not rely on spatial quantization. In [36] a subcategory-aware version of R-CNN is presented. Although the output is not exactly a segmented object region, this approach incorporates subcategory information to improve region proposals. It independently isolates objects, even if they belong to the same class, which totally answers to the main requirement of instance segmentation.
Hayder *et al.* proposed in [38] a new approach to estimate object masks exceeding the scope of RoIs estimated in advance by processing the data in the distance transform domain. The model called Object Mask Network (OMN) embraces a ResNet similar architecture trained to perform boundary-aware instance segmentation.
In [39], Van den Brand *et al.* proposed a method for separating vehicles in the context of autonomous driving scenarios. The approach is based on combining contour and object detection. Likewise, [40] showed that combining instance-agnostic semantic segmentation and instance-aware boundaries (produced by separate subnetworks using a MultiCut framework) allows for an effective instance separation. The Sequential Grouping Network (SGN) presented in [41] by Liu *et al.* is a combination of subnetworks. One of these predicts object breakpoints in vertical and horizontal directions, the second joins them to produce connected components and the third fuses the resulting components into semantically meaningful object instances. Moreover, Arnab *et al.* introduced in [42] a complex architecture with two communicating modules. The first is responsible for producing initial class-level semantic segmentation. The second uses these to perform instance separation based on CRF predictions combined with other available detection cues. Ren *et al.* proposed in [43] a slightly different architecture consisting of four major modules (external memory, box proposal network, segmentation network and the scoring network). It uses LSTM units to simulate the recurrent human attentive process allowing for distinguishing object instances. This recurrent neural network (RNN) does not holistically consider the input. It performs segmentation in an iterative way considering inherent instances one after the other like a temporal chain.
The model introduced by Uhrig *et al.* in [44] combines semantic with depth information to segment instances. The scope around object centers is quantized into a set of angle classes to define directions. Each pixel is then assigned to its corresponding instance depending on its direction against the relevant center. Similarly, [45] proved that multi-tasking training could benefit tasks jointly solved. In fact, Kendall *et al.* showed that weighting the loss terms of jointly trained tasks based on the homoscedastic likelihood considerably improves scene understanding e.g. depth estimation, class-level segmentation, instance-level segmentation...
A simple approach inspired by standard watershed algorithm was proposed by Bai *et al.* in [46]. It employs an end-to-end deep architecture to represent instances as separable energy basins. The last energy level is generally cut away to enforce the separation between independent objects. In [47], Zhang *et al.* solve the task with a model that contains

a densely connected Markov random Field (MRF). It performs instance segmentation by exploiting joint information coming from previous CNN patch-based predictions, contrast-sensitive smoothness cues as well as significant long-range connections from instances.

2.3 The Proposed Approach: MokaNet

The interesting potential of Convolutional Neural Networks (CNNs) in terms of performance, robustness and low complexity in the context of scene understanding related to autonomous driving scenarios is undeniable. For instance, standard classical approaches are completely absent among to the top-level performing methods of semantic segmentation benchmarks such as Cityscapes [1]. Obviously, the margin between DL and classical approaches is now so important that comparison is no longer fair. That is why, we propose here a pure CNN-based solution for alleviating this problem.

2.3.1 Implementation Details

Architecture

The architecture of the convolutional network is the backbone of any supervised CNN-based technique as the accuracy of the learned mapping between data and labels strongly relies on it. To this end, we developed a multi-modular architecture which best suits our application. The main used building modules are sub-networks inspired from the work we introduced in [48] as a solution to the task of initial background estimation.

Before discussing the layout of the elementary building blocks, we first take a look at the overall design of the introduced architecture. The main innovation we offer here in terms of design is the concept of "cascade". This revolves around piling a set of similar encoder-decoder networks in series in order to form a chain of successively connected sub-networks. The recurrence of sub-networks proposed by our scheme is different from the recurrence of RNN-models. The former is vertical seeking more depth of the pipeline and the latter is horizontal seeking the replication of the network in temporal dimension to parallelize the processing by using gated Long-Short Term Memory (LSTM) units. The motivation behind our suggestion is the fact that applying changes and modifications on deep architecture is a hard task. In fact, a small change in any layer can have an important effect. In order to make it easy for the designer to control this, our architecture allows him to target always a specific sub-network instead of considering the whole architecture. From the implementation point of view, this is a huge advantage considering that CNN training is a pure trial-error task. Obviously, our model offers more flexibility for hand-engineering since the different building blocks could be independently accessed. Moreover, this flexibility alleviates many further common problems faced during the training of deep architectures such as escaping local minima. Clearly, the deeper the architecture is, the higher is the probability to land in a local minimum. In fact, the optimization of highly non-convex objective functions using complex parametric models can be seen as

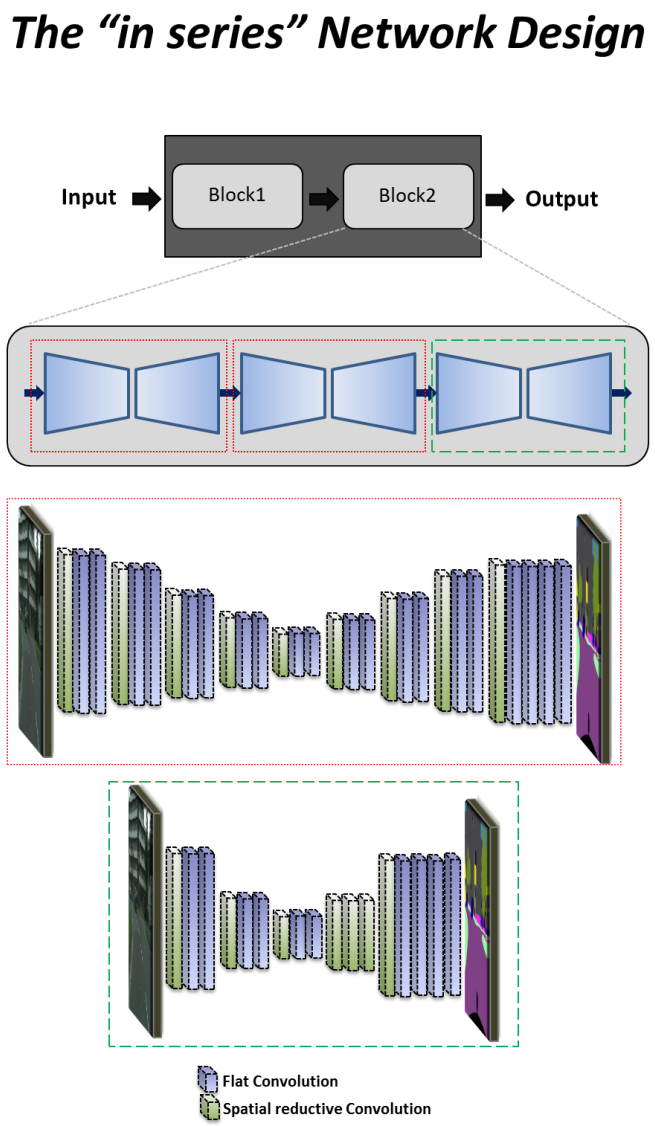


Figure 2.3: The MokaNet cascaded architecture.

an optimization problem in a very large parameter space fraught with "bad" local optima. Modifying the design is one of many possible measures that can be taken to avoid this. Moreover, the issue of gradient diffusion in deep networks could be also better tackled through the cascading scheme. In fact, the back-propagation step is based on the derivative of the cost function calculated with respect to all network's parameters. The update of these parameters is then realized according to the resulting gradient values in every iteration. If the structure is very deep, the back-propagated gradients will start to vanish

gradually at a certain level till they reach the very early layers as almost zero. Consequently, the affected layers will have their parameters hardly changed and end up not learning properly. Our model offering intermediate loss layers for each independent block is less eligible to the problem thanks to the reactivation effect realized at each of these intermediate stations. Every time the gradients start to fade during back-propagation, the loss of the next sub-network accordingly redresses them.
Taking these aspects into consideration, the final network we propose is designed like "Matryoshka dolls" (the Russian nesting dolls). It is built as a set of stacked sub-networks to form larger blocks that are themselves stacked over each other. After exhaustive experimentation and parameter tuning, the final structure looks as shown in Fig. 2.3. The different blocks are exactly similar. Each one is formed by three stacked sub-networks. The two first have 5 layers for convolution with spatial shrinking (using stride $s > 1$) in the encoder part and the same number of symmetrical transposed convolutional layers (also with stride $s > 1$) in the decoding stage. Each of the shrinking convolutions is followed by two layers of flat convolutions with stride $s = 1$ except for the final one that is followed by four ones.
Unlike the two first sub-networks, the third one is just a 3-depth paradigm. It comes with 3 layers of spatial shrinking convolutions and 3 transposed convolutional ones in encoder and decoder parts respectively. Similarly, two flat convolutions follow each spatial shrinking one except for the last layer that is followed by four.
After the overview of the whole structure, let us move on to discuss the details (5-conv sub-network as example). The design of all sub-networks was initially inspired by the FlownNetSimple architecture proposed by Dosovitskiy *et al.* in [49]. It is indeed a two-stage architecture as shown in Fig. 2.2. The encoder is composed of a succession of convolutional layers. This generic stage extracts high level abstractions of the stacked input images. It forwards the gained feature maps to the decoder stage in order to enhance the coarse-to-fine transformations. This enhancement aims at recovering the details and information lost through the different spatially shrinking convolutions. To achieve this, the low resolution feature maps are successively up-sampled to finer representations, achieving a dense per-pixel representation at the output which subsequently benefits the nature of the segmentation task. However, To fully adapt the design to the specificity of our applications, few modifications are additionally suggested. The main motivation behind the introduced changes is to improve the quality of the feature maps during both encoding and decoding stages. As shown in the final design in Fig. 2.4, this is achieved by taking advantage of the redundancy gained from other layers by using additional concatenations at each layer. This way, we combine multiple feature maps representing different high level abstractions from previous stages and containing more information. Therefore, a wider range of features is obtained which in turn enables a more efficient and faster optimization. Likewise, the enforced redundancy improves the accuracy of the coarse-to-fine step during each transposed convolution thanks to additional information. This strategy enables a better recovery of fine details lost in low-resolution feature maps. In fact, importing features from early stages where just few details were lost on processing, and adding

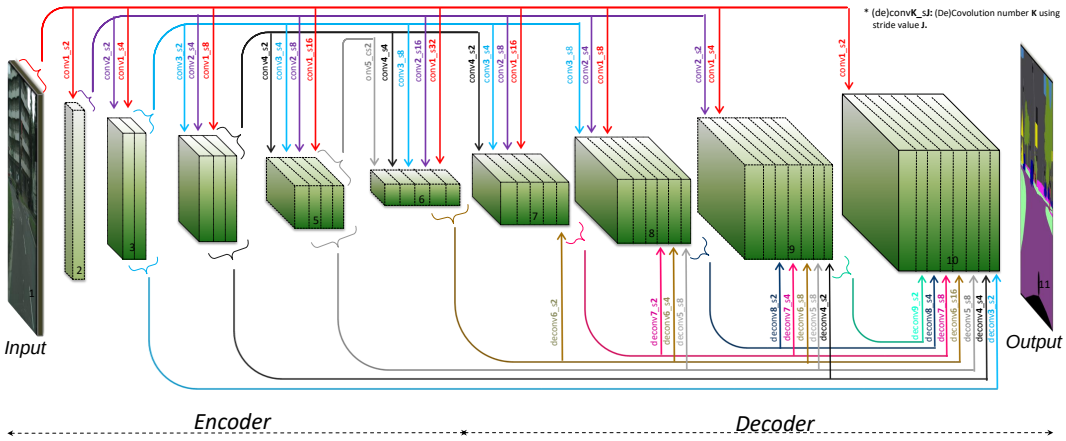


Figure 2.4: Sample architecture of a single sub-network with depth 5 (5 different resolution levels).

them to later layers has the advantage of guiding the learning process to keep track of the very fine details that are easily lost along such deep architectures. Such details are indeed crucial for high quality refinement of the segmentation. To this end, we perform multiple convolutions on each level of the encoder with varying settings (different strides). The considered strides should be power of the fixed stride value $\hat{s} = s(i; i+1)$ used between any two consecutive convolution layers i and $i+1$. For instance, we consider $m \geq n$ as the ranks of two non-successive convolutional layers in the architecture. To guarantee that m-layer output has the same dimensions as the n-layer input, the used stride is set to $s(m,n) = \hat{s}^{(m-n)}$. We concatenate the resulting feature maps to the inputs of the next layers accordingly based on the respective dimensions. Similarly, we apply several transposed convolutions on coarse feature maps from encoder and decoder and finally concatenate them accordingly to the inputs of the corresponding transposed convolutional layers.
We introduce a loss after each scale level in both parts by always bringing the ground-truth label to the same resolution of the layer's output. The final loss is a weighted sum of the different loss terms present in the architecture. The choice of the weighting factors was empirically examined. In fact, a linear correlation between the weight allocated to the intermediate loss and the number of the output feature maps given by the corresponding layer is noticed. The more features are present, the higher should be the contribution of the corresponding loss term to the final loss. The choice of the feature maps cardinality was itself empirically set up. It obeys an exponentially ascending order for the encoder part staring by 32 for the first level, 64, 256, 512 and 1024 to the other levels respectively. The same numbers are used in the opposite order for the decoder part so that symmetrical levels in terms of dimensions equally contribute to the final loss function. In fact, the deeper we go in the network the more details are eligible to be lost because of the interpolation. That is why more feature maps are needed at each scale level to improve the information encoding. However, if extremely large numbers are chosen, the size of

the network (the number of parameters to optimize) increases drastically which requires more resources for the training. Besides, a decrease in the generalization performance is expected in this case due to potential over-fitting of the large model against the used training data. Therefore, a reasonable choice of these numbers should be always made based on a trial and error strategy with respect to the available resources at hand.
Ultimately, if we compare the same architecture with and without our proposed extra-links, the former will present more parameters to learn while keeping the same number of filters for each original scale level. Although this requires more resources, it clearly increases the computational efficiency of the model and guarantees faster convergence during the training phase.
At the time of joining the different sub-networks to form the overall architecture, we interpose, as previously discussed, the layers performing the flat convolution between the intermediate layers. The goal is to encode neighboring information into each pixel without loosing further details because of striding. Together with the dense extra-links, the flat convolution results in a strongly smoothed version of the information inherent in the incoming feature maps. We have first tested these layers with low-level image processing tasks such as de-noising, super-resolution and high dynamic range imaging. They enhanced significantly the smoothing quality and the noise reduction. As we noticed that refinement is needed in the context of segmentation as well, we applied the same strategy there in order to enforce better aggregation especially during the decoding stage where information recovery is performed.

Data Augmentation

For DL applications, data is a crucial pillar of the training procedure. Quantity as well as quality are important aspects to take into consideration. In case of data shortage, it is possible to generate processed augmented data from the available set synthetically. This can be an efficient way to not only enlarge the training set but also to vary its content at the same time.
Depending on the application, various operations could be applied on the training data in order to augment it. Mainly, two types of data transformations were used to augment the original training set for semantic segmentation (in total 3475 images = train + validation sets of Cityscapes). The first category corresponds to the geometrical transformations. First, image resizing is performed to bring down the resolution to 512×256 due to memory restrictions. For rotation, translation, scaling and horizontal/vertical flipping, we use the spatial transformer network module [5] fed by the random generated affine matrix from equation. 2.1 (Max and Min values for each transformation are pre-set by the user and the values are randomly generated within the given interval during the training) to apply the transformations on the fly during the training without need to create and store data locally.

The random affine matrix generator produces a transform matrix

$$M = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \tag{2.1}$$

this is applied on points of the input image grid to transform them correspondingly. As an example, the transform of the point $P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ of an input image with normalized coordinates is the new point P' in the augmented image defined by

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.2}$$

where c and f translation coefficients,

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & f \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+c \\ y+f \end{pmatrix} \tag{2.3}$$

a and e are responsible for the scaling ratio

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & e & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ e \cdot y \end{pmatrix} \tag{2.4}$$

and a, b, c and d are used for counter-clockwise rotation by angle θ if

$$\begin{gathered} P' = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & b & 0 \\ d & e & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \\ \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot \cos(\theta) - y \cdot \sin(\theta) \\ x \cdot \sin(\theta) + y \cdot \cos(\theta) \end{pmatrix} \end{gathered} \tag{2.5}$$

The discussed first category of transformations is expected to be applied on input and ground-truth images simultaneously in order to keep the alignment. (For further details on the STN module please refer to section 4.2.1 where implementation details are elaborated).
For the second type of data augmentation called photometric, only input images are considered. In fact, just low-level characteristics of the input images are modified without content change that is why ground-truth labels are not updated. For this, we opt also for online application. The considered transformations were, Gaussian noise addition with a randomly generated variance in $[0.01, 0.05]$, contrast modification with a factor in

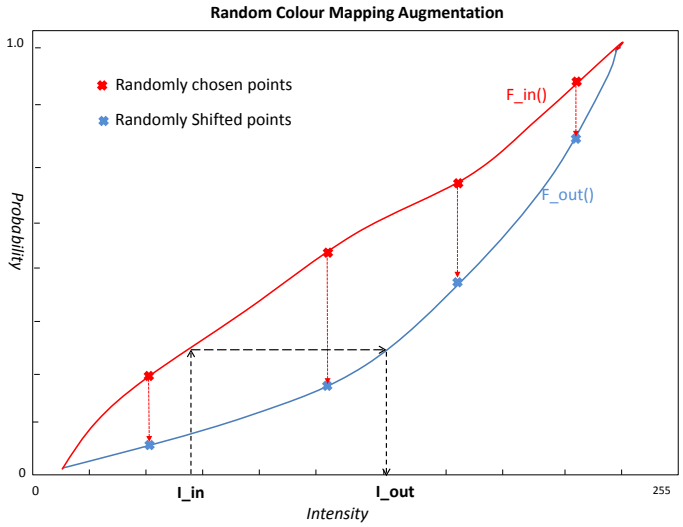
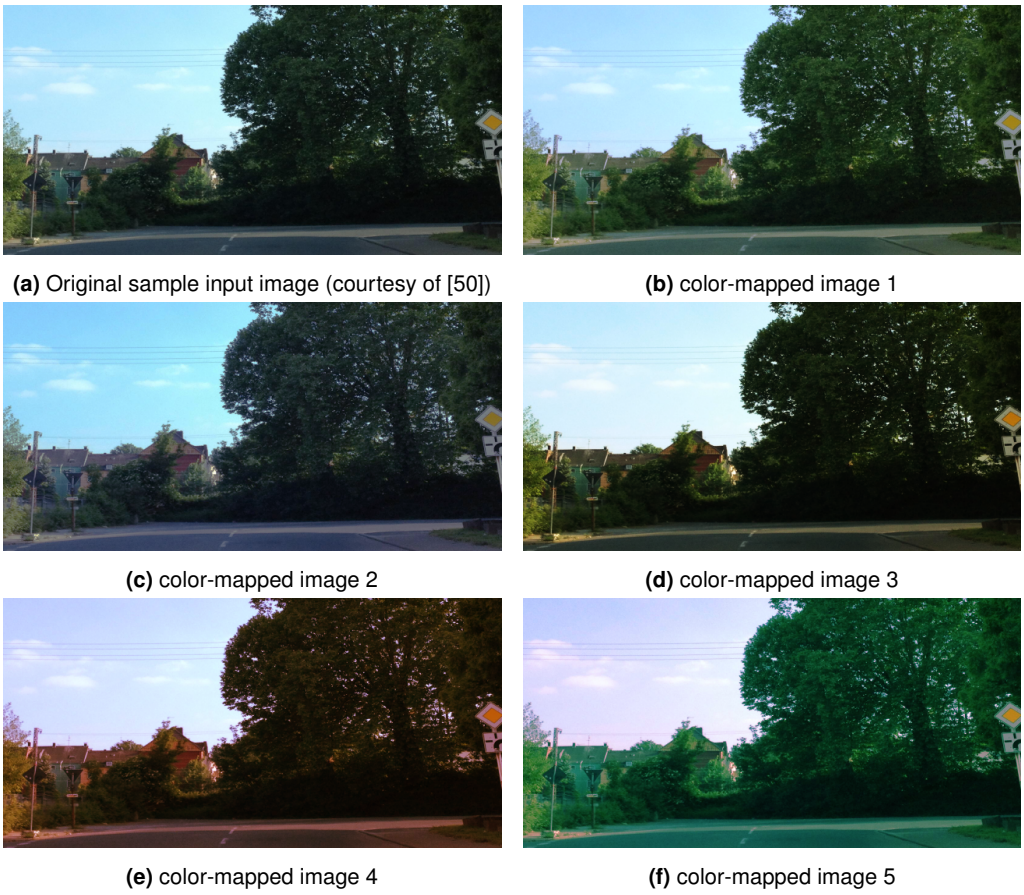


Figure 2.5: The random color-mapping image augmentation to generate images with different color ranges.

$[-0.6, 0.3]$, gamma value correction using values in $[0.5, 1]$ and brightness change with variance in $[0.1, 0.3]$.
Moreover, we use additionally an offline color-mapping augmentation in the goal to simulate different camera response functions. This varies the nature of the available images and the characteristics of the treated signals. It enhances the generalization capabilities of the model and guarantees more robustness in the case of tests images recorded by different capturing systems. For each image, we calculate its histogram and the resulting cumulative distribution function (cdf) $F_{in}()$. The expected output after the augmentation step is an output image with a different cdf $F_{out}()$ showing significant color variations. After calculating $F_{in}()$, we randomly select four intensities $I_k, k \in 1,2,3,4$ on the x-axis with their corresponding cdf values $F_{in}(I_k), k \in 1,2,3,4$ lying on the curve of $F_{in}()$. Then, we randomly shift the resulting points along the y-axis within a specified range $[F_{in}(I_k) - 0.5, F_{in}(I_k) + 0.5]$. Based on the resulting new points $F_{out}(I_k), k \in 1,2,3,4$, we create the new cdf curve $F_{out}()$ belonging to the expected transformed image using the Bezier interpolation. After getting the cdf of the target image, we map every single intensity based on the difference between the two curves. Let us assume we have input intensity I_{in}, we calculate its image $F_{in}(I_{in})$, we project this value on the output cdf curve and we project back the resulting value on the x-axis in order to get the mapped intensity I_{out} as shown in Fig. 2.5. For each image in the training set, we randomly generate color mapped versions following the previous procedure in order to enlarge the original training dataset (example as in Fig. 2.6).
All discussed data transformations (geometric and photometric) are combined together and moderately applied on available images to enlarge the original set and enrich its



(a) Original sample input image (courtesy of [50])

(b) color-mapped image 1

(c) color-mapped image 2

(d) color-mapped image 3

(e) color-mapped image 4

(f) color-mapped image 5

Figure 2.6: Sample outputs of the color-mapping augmentation.

entropy without corrupting the content by causing extreme changes.

2.3.2 Experiments & Results

Training details

For all our experiments, we use the "cascaded" architecture, previously detailed, to perform the training. After a thorough experimentation phase, we decided to employ a custom version of Caffe [51] and make use of the Adam optimizer [52] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. This optimizer, as defined by the authors, "is a gradient-based first-order optimization approach that proved to be well performing and efficient for solving non-stationary cost functions with noisy and/or sparse gradients. Besides, it is well suited for problems that are large in terms of data and parameters and its hyper-parameters have intuitive in-

terpretations and typically require little tuning".
The training is realized with batches of size 2 till the convergence, generally around $3.5M$ iterations. The starting learning rate $\lambda = 10^{-4}$ is progressively multiplied by a factor of 0.5 every $150K$ iterations. The used decay factor for the weights is $\delta = 10^{-4}$ and the gamma value is $\gamma = 0.5$.

Class-level Semantic Results

As a starting point, we focus on performing class-level semantic segmentation in autonomous driving scenarios using the previously discussed setup. For evaluation, we consider the semantic segmentation benchmark of Cityscapes [1] as it offers the most detailed ranking of the best performing DL models for urban scenes semantic segmentation.
Cityscapes presents a large scale stereo-vision dataset for semantic understanding of urban scenes recorded under different conditions in various cities around Germany. It offers 20000 weakly annotated images with coarse semantic segmentation labeling and 5000 images with high quality fine-annotated ground-truth. We focus rather on the fine-labeled set of images that consists of 2975 frame for training, 500 frame for validation and 1525 frame for test. Up to 30 different classes categorized into 6 large object categories are considered for labeling Cityscapes' segmentation ground-truth images (For detailed overview please refer to the dataset labeling policy [1]). For the sake of simplicity, only the most relevant and reasonably frequent traffic object classes (19 classes), from the application point of view, are considered for evaluation and benchmarking.
Considering the available resources, training on images in their native resolution $W \times H = 2048 \times 1024$ is not possible. Therefore, we opt for resizing images and ground-truth labels down to $W_{crop} \times H_{crop} = 512 \times 256$ by keeping the same aspect ratio. We apply the previously discussed data augmentations by adding 4 additional color-mapped images for each frame to enlarge the set to 11900 images. Together with the online augmentations, previously discussed, we train our model up to $3.5M$ iterations.
Once converged, the final model is used to generate results on images from the validation set shown in Fig. 2.7 together with their given ground-truth labels and on images from the test set as shown in Fig. 2.8 where no given ground-truth is available. The images shown in Fig. 2.9 depict scenes recorded in China with a totally new landscape, weather conditions, road signs to our model and the corresponding estimated output segmentation masks.
Aside from the visual assessment, we evaluate our results numerically on the validation set of Cityscapes that consists of 500 resized to 512×256 with their corresponding semantic segmentation ground-truth labels. Results in Tab. 2.1, present a detailed numerical evaluation of our model in terms of Intersection over Union (IoU) for each class separately and the inter-class intersection over union (iIoU) averaged over the whole validation set. The PASCAL VOC intersection over Union "IoU" measure [53] is an evaluation metric for measuring the semantic segmentation accuracy against a pre-given ground-truth mask. It is a ratio between two terms: The first term in the numerator corresponds to the area of

overlap between the predicted class region and the region depicting the same class in the ground-truth mask. The second term is the denominator and describes the union area of the two same regions. The division of the two terms quantifies the similarity between the estimate and ground-truth. It is written as:

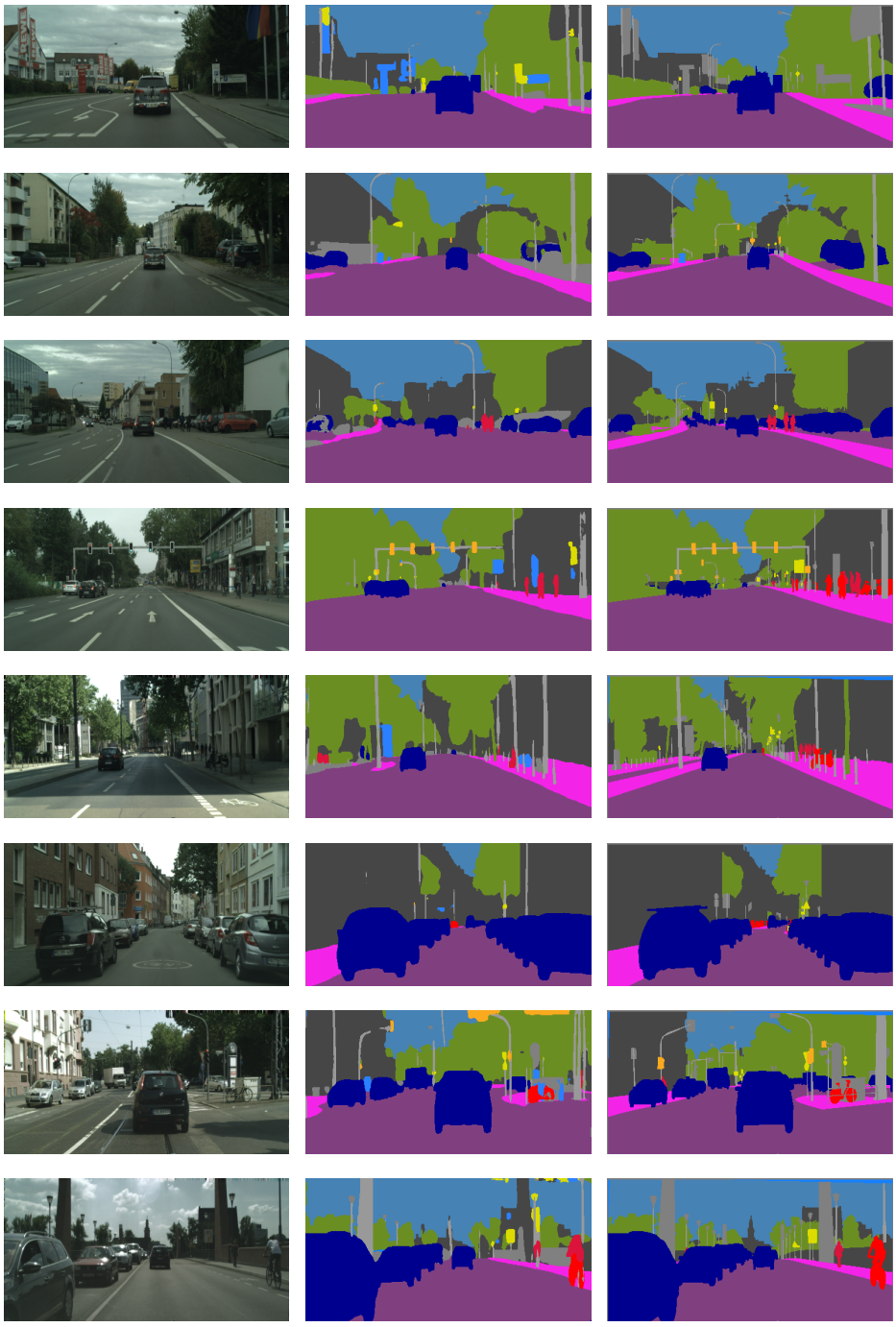
$$IoU = \frac{area(B_{est} \cap B_{gt})}{area(B_{est} \cup B_{gt})} = \frac{TP}{TP + FP + FN} \tag{2.6}$$

where, TP, FP, and FN denote the true positive, false positive and false negative counts respectively. B_{est} is the estimated area depicting a certain class and B_{gt} is the class area in ground-truth label.
However as addressed in [54], it is known that the IoU measure is biased toward classes with instances that occupy large image areas. In the case of strong scale variations, this problem could corrupt the evaluation reliability for key classes such as for cars in our case. An additional metric called instance-level Intersection over Union "iIoU" [54] is proposed to evaluate the robustness of the segmentation accuracy against scale variations. Similarly,

$$iIoU = \frac{iTP}{iTP + FP + iFN} \tag{2.7}$$

where FP is again the number of false positives and iTP and iFN are numbers of true positive and false negative pixels respectively. iIoU computes the weighted contribution of a every single pixel multiplied by the ratio of the class' average instance with respect to the size of the ground truth instance. Obviously, this measure is not computable for all classes. Only classes with instance-level labeling in the ground-truth are considered.

In Fig. 2.7, visual quality of the output estimated semantic segmentation mask of our designed model could be compared to the given ground-truth semantic segmentation labels. Exclusively, only validation set images are considered. The model proves highly performant on scenes recorded in different cities under various conditions despite being trained on different images captured around other cities in Germany. This proves the generalization capability of the network on German traffic scenes and its ability to depict the finest details including small, far and dark traffic signs (e.g. rows 3, 7 and 8) that are barely visible with human eye and fine poles (e.g. row 1, 7 and 5) hard to detect. Moreover, we denote the high quality segments of the important traffic component classes, namely cars that represent our first targeted class-object for this project. In fact, results show that cars are properly segmented (considering the given ground-truth) regardless differences in colors, forms, models, distance and covered area in the image.
Previous observations can be confirmed by results displayed in Fig. 2.8 where a set of images recorded in a third different set of German cities is considered. These images selected from the test set of Cityscapes come indeed without publicly available ground-truth to avoid potential parameter fine-tuning. As noticed, MokaNet performs well on the test images. Car objects are equally well segmented and the segmentation quality of small far objects is just as high as in the previous set. This gives a further proof about the model's



(a) Input frame

(b) Estimated semantic segmentation mask

(c) Ground-truth semantic segmentation label

Figure 2.7: Visual Evaluation on Cityscapes Validation Set: (a) Example frame selected from the validation set of Cityscapes [1] (b) The estimated semantic segmentation (c) Ground-truth semantic segmentation.

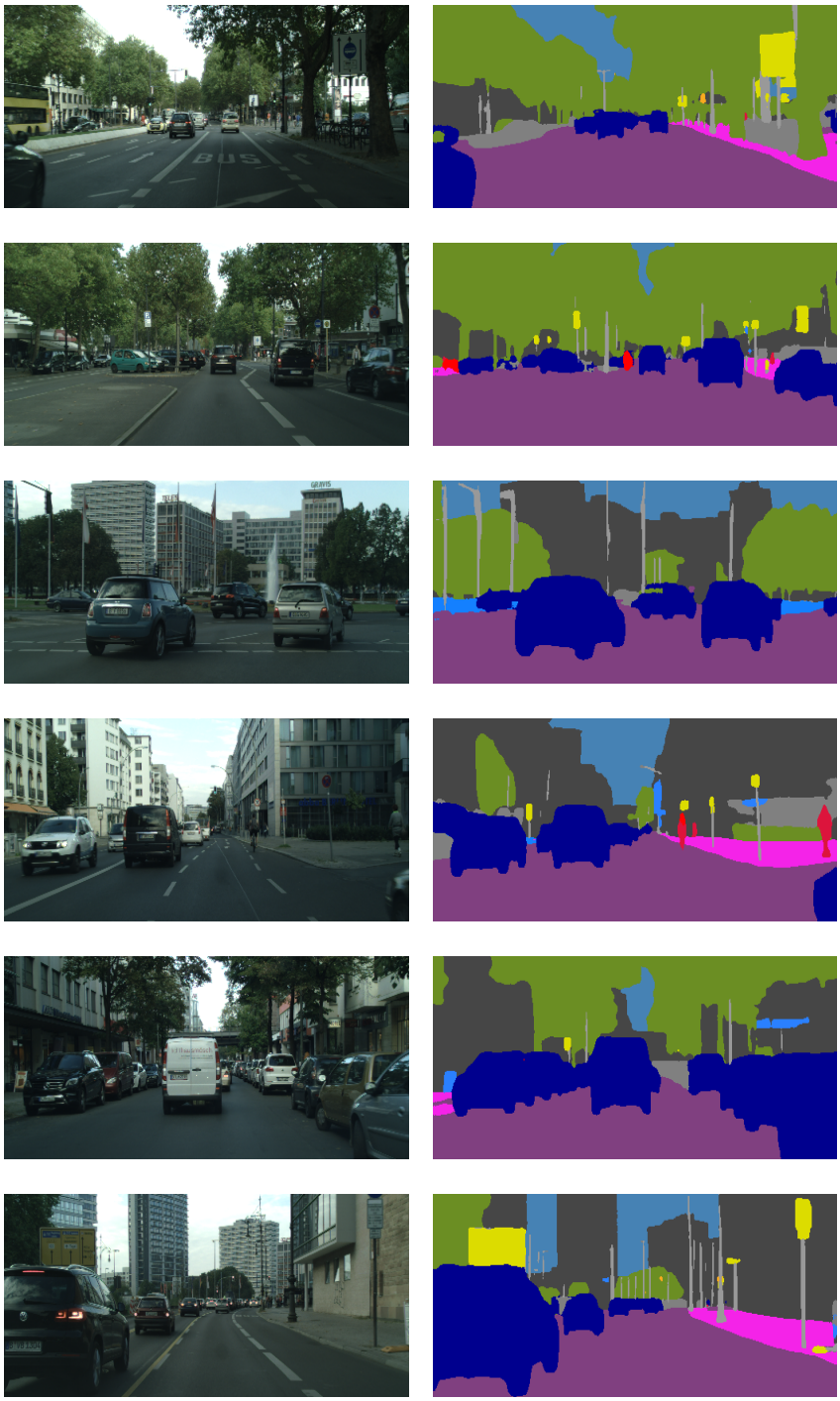
robustness and the transparency of the used training procedure.
In the same regard, MokaNet has been challenged to perform on further images recorded in total different environments. These present traffic conditions and characteristics that are significantly different from German traffic. In fact, estimated results on exclusive images recorded by our colleagues in China are shown in Fig. 2.9. The results prove that our model delivers high quality semantic segmentation despite the challenging scenario of a totally different semantic context. This observation highlights the fact that the used data augmentation strategy is considerably enriching and diversifying the training set. It significantly improved the model robustness against new agents and unexpected semantic contexts. Consequently, MokaNet is able to properly segment Chinese traffic scenes despite the undeniable differences (the kind of vegetation, building style, the traffic signs' language and shape, the vehicles' models, shapes and forms).

In order to properly assess the MokaNet performance, we present in follows a quantitative evaluation of the Model's performance on the 500 images of the validation set of Cityscapes with resolution 512×256 using the pre-discussed IoU (Equation. 2.6) and iIoU (Equation. 2.7) measures. Tab. 2.1 presents a detailed survey of the quantitative results for each single object class (19 class). IoU is computed for all classes whereas iIoU is only relevant for dynamic ones. Obviously, the most important traffic components especially cars show high accuracy values using both measures. This can be related to the high frequent occurrence of objects belonging to this class in the dataset and also to the important number of pixels belonging to such objects relatively to the whole image. Under these conditions, the learning process profits and the network is able to learn more features of this specific class during training. This also explains the difference in terms of values between the different classes. The more the network sees a certain class (in terms of frequency and region size), the better is the learning output for this one.

If we consider the performance on the level of object categories (as proposed by the Cityscapes labeling policy), then we get even better numerical evaluation since high frequent classes make the balance with less frequent ones within the same category. Results are reported in in Tab. 2.2. The vehicle category still has high accuracy using both IoU and iIoU just behind the flat (road, sidewalk, parking, etc...) category which offers obviously more learnable material (it is frequent in terms of occurrence and mostly occupy an important region of each traffic scene).

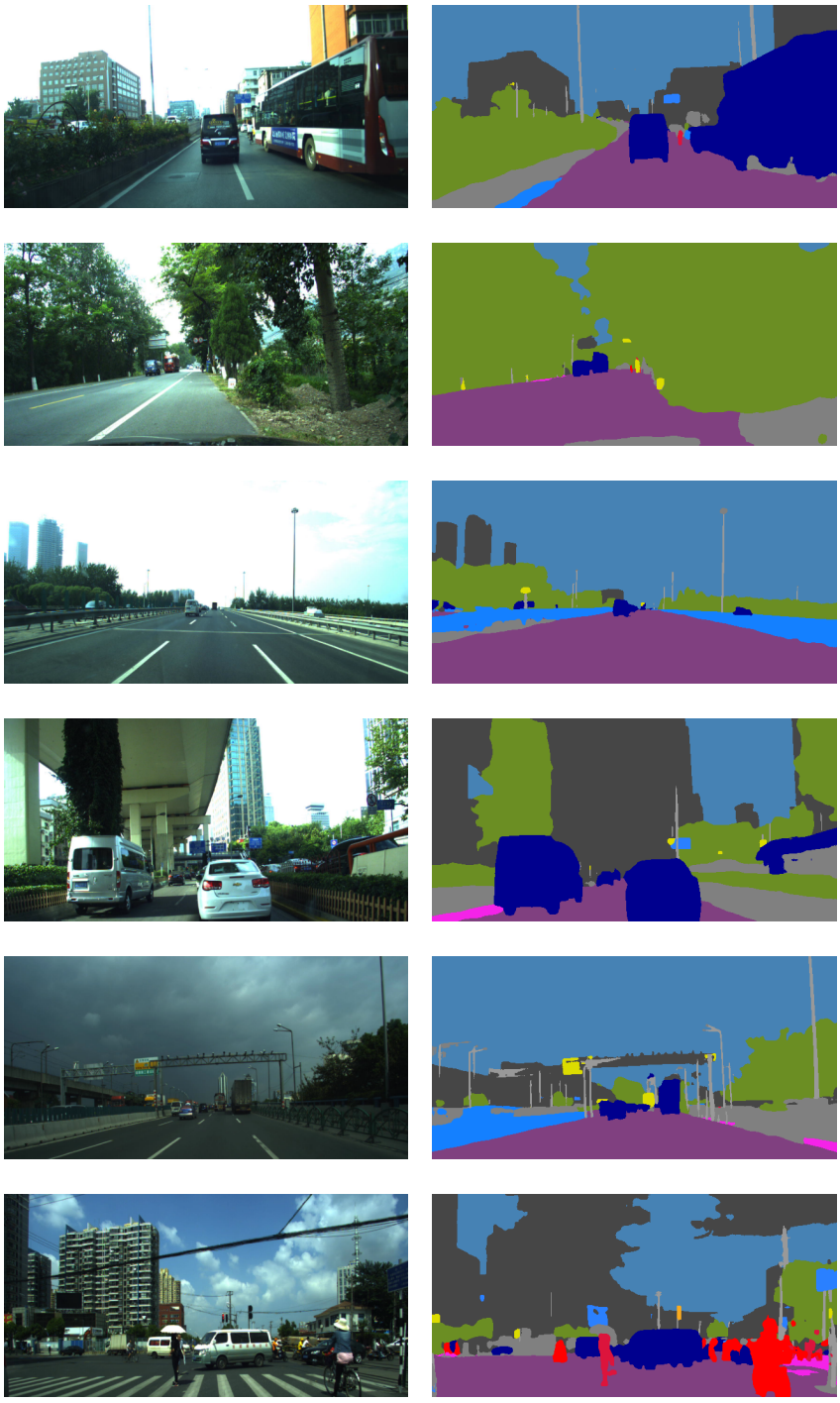
Since vehicles and cars are highly prioritized in the context of the project, we specifically examine the models' performance for their accuracy. Tab. 2.3 shows the numerical results. In fact, MokaNet outperforms all remaining models in terms of car segmentation accuracy across the whole validation set. Resulting output images displayed in Fig. 2.10 prove the same observation in terms of visual quality. Clearly, MokaNet is the best alternative for autonomous driving semantic segmentation in comparison to other state-of-the-art DL models with publicly available code.

In both tables Tab. 2.3 and Tab. 2.4, we extend our performance analysis by a quantitative survey of best performing state-of-the-art CNN models for semantic segmentation that has been trained and tested under the same configuration setup. This includes the training



(a) Input frame (b) Estimated semantic segmentation mask

Figure 2.8: Visual Evaluation on Cityscapes Test Set: (a) Example frame selected from the test set of Cityscapes [1] (b) The estimated semantic segmentation



(a) Input frame (b) Estimated semantic segmentation mask

Figure 2.9: Visual Evaluation on China Images: (a) Example frames recorded in China (b) The estimated semantic segmentation

Class-object	IoU	iIoU
Road	0.975	—
Sidewalk	0.813	—
Building	0.887	—
Wall	0.502	—
Fence	0.479	—
Pole	0.484	—
Traffic light	0.403	—
Traffic sign	0.549	—
Vegetation	0.899	—
Terrain	0.706	—
Sky	0.904	—
Person	0.641	0.471
Rider	0.427	0.277
Car	**0.897**	**0.856**
Truck	0.748	0.397
Bus	0.779	0.507
Train	0.663	0.341
Motorcycle	0.362	0.119
Bicycle	0.542	0.484
Average	0.666	0.431

Table 2.1: Performance assessment for class-based semantic segmentation on the validation set (500 image) of Cityscapes [1] with resolution (512×256) using MokaNet.

Categories	IoU	iIoU
Flat	0.979	—
Nature	0.900	—
Object	0.524	—
Vehicle	**0.869**	**0.837**
Sky	0.904	—
Construction	0.888	—
Human	0.658	0.493
Average	0.817	0.665

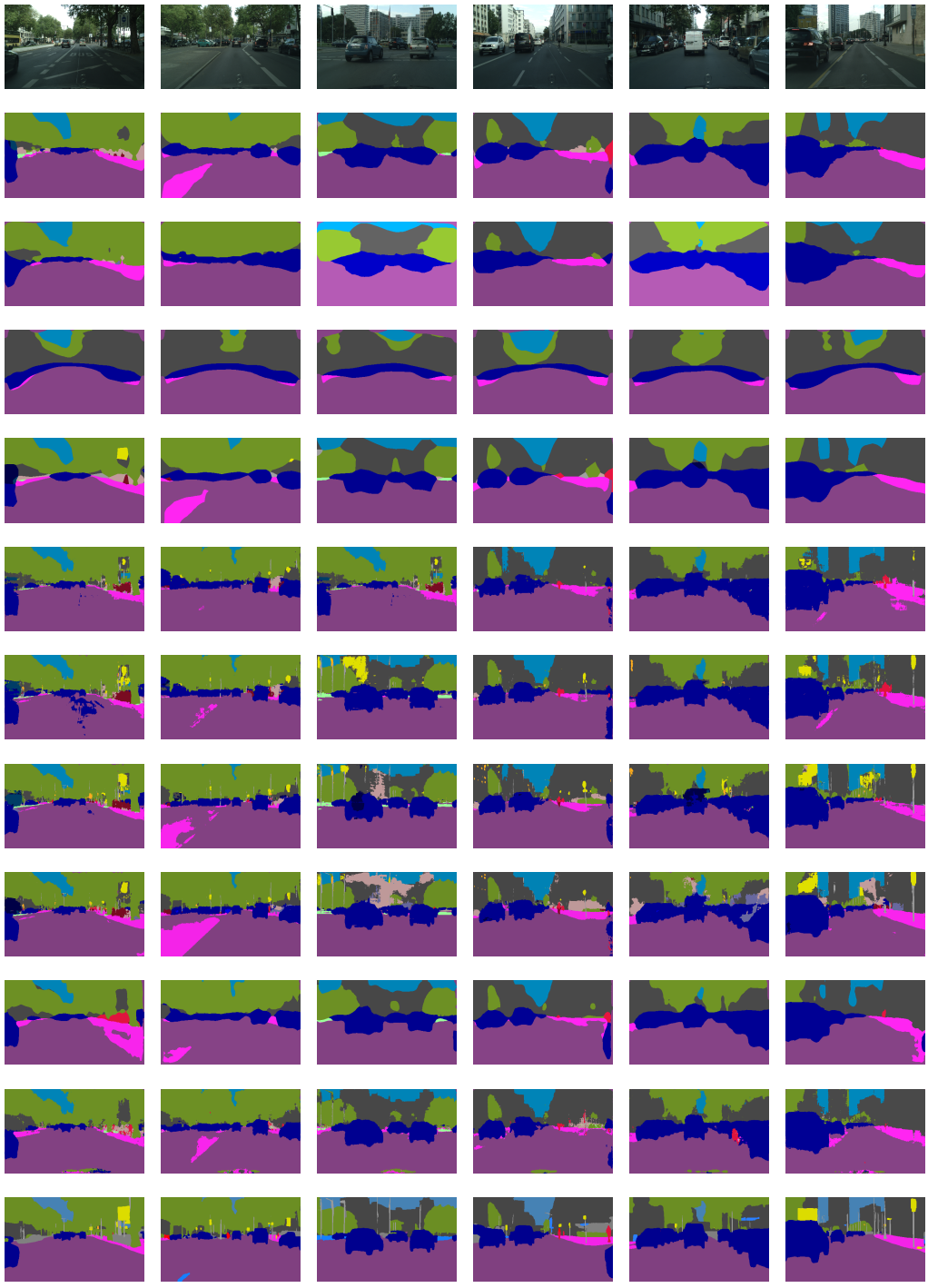
Table 2.2: Performance assessment for category-based semantic segmentation on the validation set (500 image) of Cityscapes [1] with resolution (512×256) using MokaNet.

Method	IoU_{class}	$iIoU_{class}$	$IoU_{category}$	$iIoU_{category}$	Runtime[s]
Ours	**0.666**	**0.431**	**0.817**	**0.665**	0.06
SegNet	0.292	0.135	0.648	0.500	0.06
FrrnA	0.410	0.255	0.699	0.509	0.07
IceNet	0.114	0.049	0.312	0.190	**0.04**
FrrnB	0.419	0.284	0.733	0.561	0.07
FCN32s	0.405	0.205	0.633	0.420	0.08
FCN16s	0.324	0.231	0.589	0.357	0.06
FCN8s	0.358	0.161	0.604	0.369	0.08
LinkNet	0.270	0.124	0.596	0.453	0.07
UNet	0.247	0.231	0.507	0.488	0.07
PspNet	0.642	0.417	0.803	0.641	0.07
ResNet	0.623	0.408	0.765	0.628	0.06

Table 2.3: State-of-the-art comparison of semantic segmentation (averaged metrics across all classes) CNN-based methods on the validation set (500 image) of Cityscapes [1] with resolution 512×256 (Implementation as courtesy of [3]).

Method	IoU_{car}	$iIoU_{car}$	$IoU_{vehicle}$	$iIoU_{vehicle}$	Runtime[s]
Ours	**0.897**	**0.856**	**0.869**	**0.837**	0.06
SegNet	0.800	0.736	0.724	0.647	0.06
FrrnA	0.808	0.737	0.757	0.758	0.07
IceNet	0.198	0.392	0.216	0.379	**0.04**
FrrnB	0.788	0.739	0.777	0.836	0.07
FCN32s	0.761	0.662	0.735	0.650	0.08
FCN16s	0.704	0.603	0.688	0.566	0.06
FCN8s	0.738	0.616	0.712	0.601	0.08
LinkNet	0.674	0.650	0.642	0.584	0.07
UNet	0.370	0.564	0.596	0.553	0.07
PspNet	0.871	0.812	0.858	0.805	0.07
ResNet	0.852	0.813	0.812	0.793	0.06

Table 2.4: State-of-the-art comparison of semantic segmentation (Only car class and vehicle category) CNN-based methods on the validation set (500 image) of Cityscapes [1] with resolution 512×256 (Implementation as courtesy of [3]).



(a) 1.Scene (b) 2.Scene (c) 3.Scene (d) 4.Scene (e) 5.Scene (f) 6.Scene

Figure 2.10: Visual comparison of state-of-the art CNN models for semantic segmentation (Implementation as courtesy of [3]) . From top row to bottom: 1. input frames 2. FCN8s' output 3. FCN16s' output 4. ICNet' output 5. FCN32s' output 6. Unet' output 7. LinkNet' output 8. FrrnA' output 9. FrrnB' output 10. PspNet' output 11. ResNet' output 12. Ours' output

conditions such as the used set and the introduced data augmentations, and the testing conditions such as the considered validation set, image resolution and framework.
Although, most models show more or less a comparable runtime performance of the order of few milliseconds (50..70ms) for a single image in resolution 512×256, we note that ioU and iIoU results averaged across all classes and over the whole validation set (500 images) show important differences. MokaNet indeed presents the best ioU and iIoU values in comparison to other models in both the 19-class and the 6-category setups.

Instance-level Semantic Results

Like the class-level semantic segmentation, we consider the benchmark of Cityscapes [1] for evaluation of the different models we propose for instance-level segmentation. In this part, we restrict our focus on the car class as it represents the most significant class within the scope of the project. The goal is to have proper separation between car entities. The same proposed solution for cars can be extended over the remaining classes. However for the sake of simplicity, we don't cover this in our results discussion.
In the following, we list the different methods we tested for this goal, namely segmenting different cars belonging to the same semantic class into independent instances. For evaluation, the AP and $AP_{50\%}$ measures introduced and well detailed in [55] and [56] have been considered as main evaluation measures. AP is defined as the average precision for a class divided across a pre-fixed range of overlap thresholds (50%,..., 90%) with a step of 5% (used in the goal to minimize bias effect towards a specific value). In other words, it is the IoU value corresponding to a single instance. $AP_{50\%}$ is the specific value for an overlap threshold of 50%.

Mahalanobis-Distance Heat Maps Inspired by the watershed-based solution proposed in [46] that consists of dividing objects belonging to the same semantic class into energy basins in order to separate the instances, we propose hereby a model able to divide each object into different levels based on the corresponding Mahalanobis distance of pixels against their corresponding centroid. The labels generation procedure for this idea starts with getting the mask of every car separately based on the instance ground-truth information given by Cityscapes as polygon coordinates. The corresponding centroid of each vehicle is defined at location $\overrightarrow{V_c} = (x_c, y_c)$. The Mahalanobis distance of each pixel belonging to that vehicle is computed against the centroid. The first step is to select all N pixel locations $\overrightarrow{V_i} = (x_i, y_i)$, $i = 1..N$ belonging to a single object mask. The centroid $\overrightarrow{V_c}$ calculated as a mean of all pixel locations is computed as:

$$\overrightarrow{V_c} = (x_c, y_c) \quad \text{with} \quad x_c = \frac{1}{N-1} \sum_{i=1}^{N} x_i, \quad y_c = \sum_{i=1}^{N} y_i \qquad \forall i = 1..N, \tag{2.8}$$

The corresponding covariance matrix for this set of pixel locations of the same car is given by:

$$S = \frac{1}{N-1}\sum_{i=1}^{N}(\overrightarrow{V_i} - \overrightarrow{V_c})^T(\overrightarrow{V_i} - \overrightarrow{V_c}) \quad \forall i = 1..N, \tag{2.9}$$

and the final Mahalanobis distance measure could be computed as:

$$D_M(\overrightarrow{V_i}, \overrightarrow{V_c}) = \sqrt{(\overrightarrow{V_i} - \overrightarrow{V_c})S^{-1}(\overrightarrow{V_i} - \overrightarrow{V_c})^T} \quad \forall i = 1..N, \tag{2.10}$$

We iterate over all cars present in each image of the training set. The minimum and maximum distances across all the dataset are computed and the range between both is divided into 20 equal intervals labeled correspondingly with specific ID-labels. We iterate again over all vehicles, we check the distance of each pixel and we label it according to its nearest interval. As shown in Fig. 2.11, the produced ground-truth label depicts only vehicles present in the image as layered heatmaps of the Mahalanobis distance range against the centroids.

(a) Input image **(b)** Generated ground-truth image

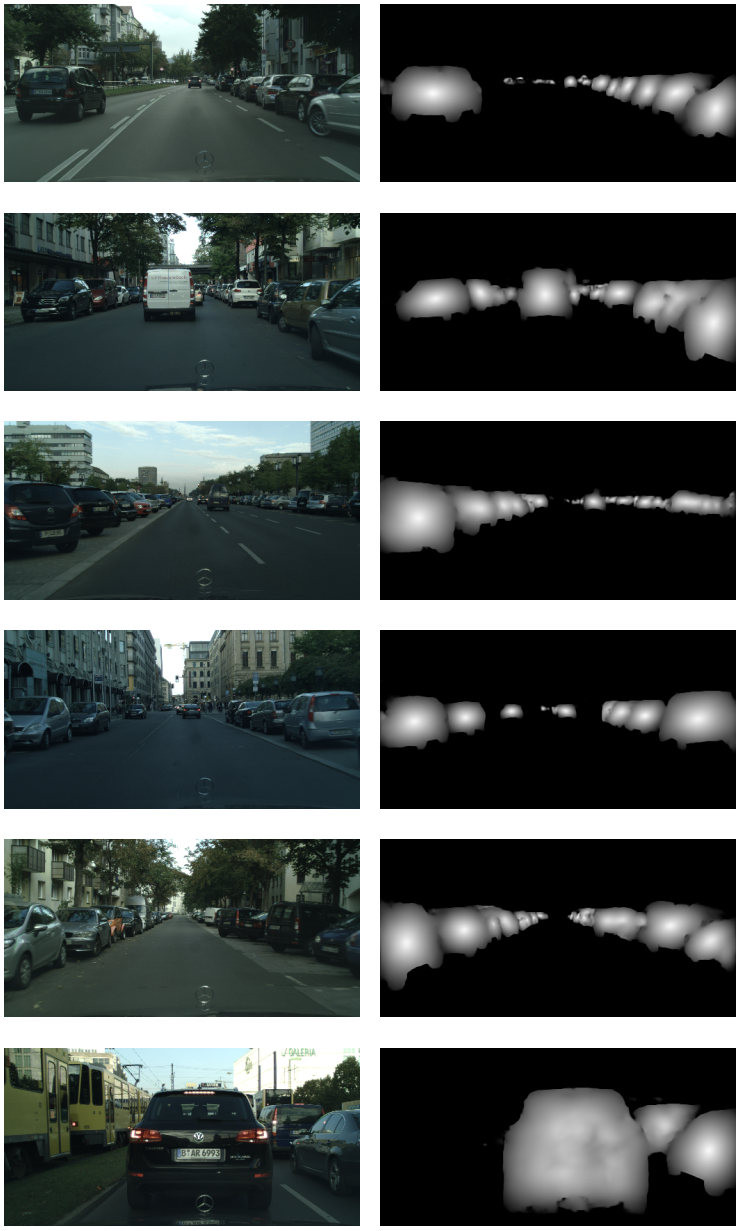
Figure 2.11: The Generation of ground-truth labels for car instance separation based on Mahalanobis heat maps.

After generating the ground-truth labels with distance heat maps for all 2975 input images of the Cityscapes training set, we apply resizing to $W_{crop} \times H_{crop} = 512 \times 256$ resolution. The same data augmentations previously discussed are applied in order to enlarge the training set by adding 4 additional color-mapped images for each frame together with the transformations applied on the fly during the training. A model with the same architecture discussed previously is then trained up to $3.5M$ iterations.

Class	AP	$AP_{50\%}$
Cars	8.0	21.0

Table 2.5: Numerical evaluation of car instance separation using Mahalanobis heat maps on the validation set (500 image) of Cityscapes [1] with resolution (512×256).

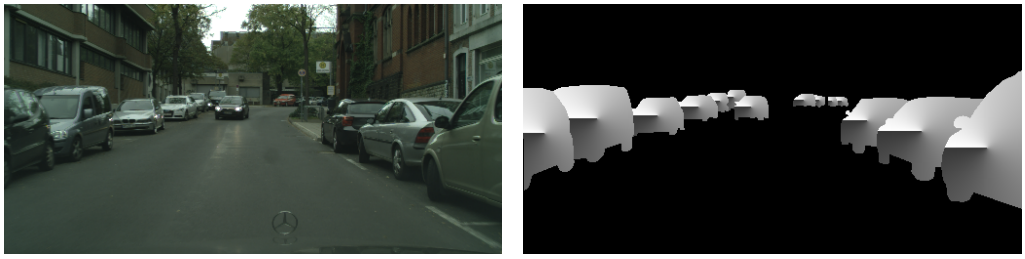
Numerical and Visual results are detailed in Tab. 2.5 and displayed in Fig. 2.12. Results can be significantly improved that is why a new idea based on angular heat maps has been used to extend the current model in the target to enhance the accuracy.



(a) Input frame (b) Estimated output mask

Figure 2.12: Visual Evaluation on Cityscapes Test Set: (a) Example frames from test set (b) The estimated output mask with the cars presented as heat maps based on the Mahalanobis distance

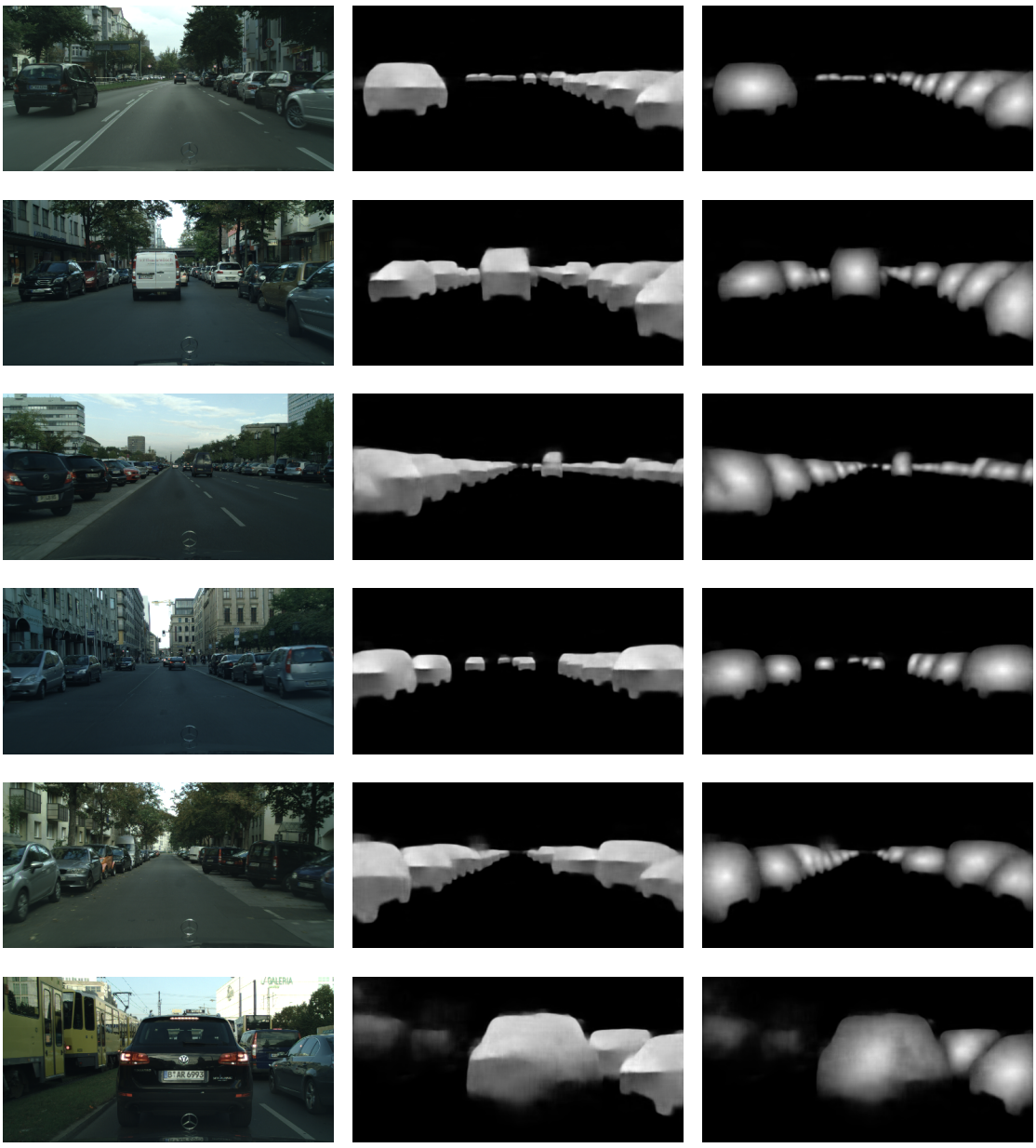
Angular Heat Maps Representing the car instances as distance heat maps can encompass meaningful information about the orientation of the vector between the pixel and the centroid thanks to the covariance matrix inherent in the Mahalanobis equation. 2.10. However, a precise orientation can be more helpful for applying instance separation between the cars.
Therefore, we decide to produce similar angular heat maps representing exclusively the orientation range instead of the Mahalanobis distance. Likewise, we segment the angular range around the center into independent intervals after setting up the clockwise direction as our conventional processing direction. We consider the angle formed by each vector relating a pixel belonging to a car with its corresponding centroid against the horizontal plane (starting from degree 0). We label it according to its nearest interval (see example in Fig. 2.13).
In association with the Mahalanobis distance, this information about the angular orientation can help better decide to which car a certain pixel belongs especially if it lies in the area between two different centroids. We also considered the assumption that a model getting both information types simultaneously can learn much more effectively to assign each pixel to the corresponding car instance based on distance and orientation.
The exact same training set up is used (architecture, data augmentations, training parameters, etc...). The only difference is that each input image to the network is now given a pair of ground-truth labels instead of one. The expected output is a pair of images with heat maps for distance and orientation. A Euclidean $L2$-norm Loss function is minimized for both outputs. The estimated results on test set images are shown in Fig. 2.14



(a) Input image **(b)** Generated ground-truth image

Figure 2.13: The Generation of ground-truth labels for car instance separation based on angular heat maps

As shown by the results in Tab. 2.6 and Fig. 2.14, the double output delivered by the new model requires an additional processing step. Information extracted from both outputs needs to be combined together in order to properly assign a given pixel to the corresponding car instance. Since an end-to-end car instance segmentation is required, alternative methods harnessing the car shapes and forms are to be explored.



(a) Input frame

(b) Estimated output mask for angular heat maps

(c) Estimated output mask for distance heat maps

Figure 2.14: Visual Evaluation on Cityscapes Test Set: Car instance segmentation based on both distance and angular heat maps

Class	AP	$AP_{50\%}$
Cars	9.0	26.0

Table 2.6: Numerical evaluation of car instance separation using distance and angular heat maps on the validation set (500 image) of Cityscapes [1] with resolution (512×256).

Concentric Contours In order to assure an end-to-end car instance segmentation, additional processing steps after the model application are to be avoided. That is why, the previous solution seems not relevant to the project requirements. As an alternative, we opt for a different direction based on defining the independent car instances as a concentric representation of the contours around their according centroids. In other words, we represent each car as a layered structure of concentric scaled versions of itself. Contours of all these versions are centered around the same original car centroid. However, we decide to divide the range into just two intervals and use only two levels of contours in order to simplify the problem to binary classification. The first level is the car scaled around the original centroid at a factor of $f = 0.95$. This is labeled as the first class. The second class consists of all pixels belonging to the range between the contour of the first class and the original contour at level $f = 1$.
For generating the new ground-truth labels of the Cityscapes training set, we proceed as before. We start with computing the centroids of all car instances through the complete training set based on the given polygon points (see Equation. 2.8). Then, each car instance is scaled around its centroid with a factor of $f = 0.95$. All pixels belonging to the scaled version are labeled as first class whereas all remaining pixels of the original object mask are labeled as second. An example of the produced output ground-truth labels is shown in Fig. 2.15

(a) Input image **(b)** Generated ground-truth image

Figure 2.15: The generation of ground-truth labels for car instance separation based on concentric contours.

Similarly, the training procedure is absolved following the same setup as in the previous experiments. Results on the Cityscapes test images are shown in Fig. 2.16.

As an exploration step, we reproduce the training of two different models where one of the classes is each time considered as background. The motivation is to examine if this difference affects the car instance segmentation accuracy. Visual results from Fig. 2.17 show

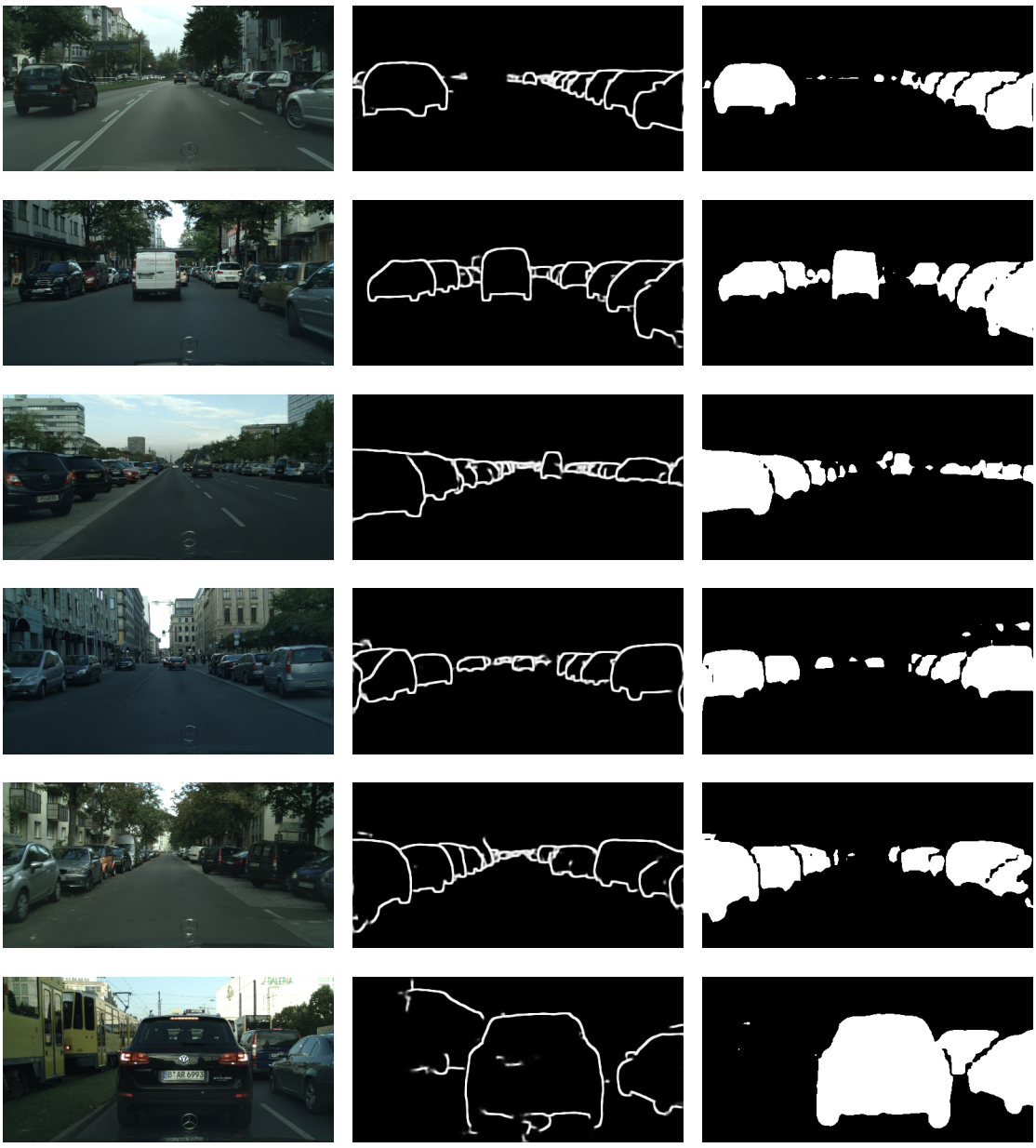
(a) Input frame (b) Estimated output mask

Figure 2.16: Visual Evaluation on Cityscapes Test Set: (a) Example frames from test set (b) The estimated output mask with the cars presented as concentric contours.

the estimated outputs for each model respectively, while Tab. 2.7 details the numerical accuracy comparison between both models. For the first one, the inner class is considered as background, and in the second the outer range is labeled as such. Obviously, performance comparison reveals that learning the outer range as foreground is slightly more efficient for instance separation. In this case the outer range of the cars includes the edges. Gradient changes are then more important and subsequently easier to identify, learn and encode as high features through the encoding part of the used architecture.

Model	Class	AP	$AP_{50\%}$
Model 1	Cars	12.0	31.0
Model 2	Cars	11.0	30.0

Table 2.7: Numerical evaluation of car instance separation using concentric contours on the validation set (500 image) of Cityscapes [1] with resolution (512×256).

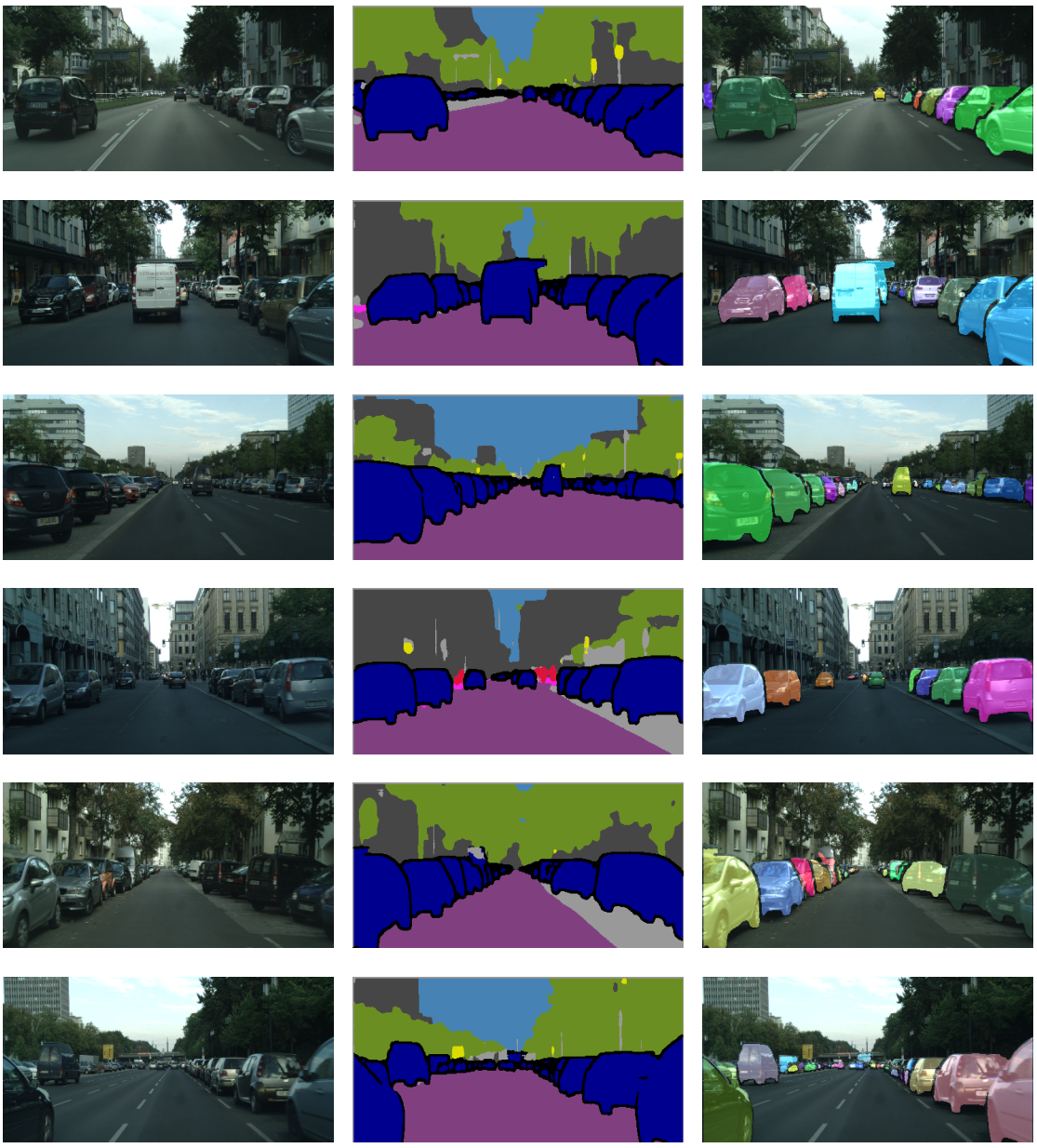


(a) Input frame (b) Estimated output model-1 (c) Estimated output mask model-2

Figure 2.17: Visual Evaluation on Cityscapes Test Set: Car instance segmentation based concentric contours differently labeled. Model-1 inner range is background and Model-2 the outer range is background.

Semantic Segmentation With Blended Car Edges Referring to the observations gained through the previous experiment, we focus specifically on harnessing the car semantic edges in order to separate the car instances. Edges, where gradient changes are more important, prove to be better learnable features to our architecture. We rely as well on the fact that this same architecture proved efficient in terms of class-level semantic segmentation. Therefore, we propose to combine both aspects to solve the instance segmentation problem by considering the fullest potential of the architecture at both experiments.
Although this new idea about blending car edges into semantic segmentation labels seems straight forward, the challenge resides in the generation of ground-truth labels. We first consider the semantic segmentation ground-truth already available. Then, we process car instance polygons provided by the dataset to create labels containing semantic edges for all cars. Both images are correspondingly blended into each other. An additional new class for car edges with an independent ID is added to the semantic segmentation ground-truth. In other words, all semantic classes are well preserved except for the car class that is segmented into disconnected instances separated by an additional new class called "car edges". The training procedure using the newly generated ground-truth remains a classification task where just the number of classes has incremented in comparison to the experiment about semantic segmentation. This way, we manage to have a proper semantic segmentation of the input image depicting a urban scene and additionally a proper separation of the inherent car instances at the same time.
After generating the ground-truth labels, we apply the same data augmentation used in the previous experiments for training. Once converged, the model is again tested on random test images of Cityscapes (results are shown in Fig. 2.18). The estimated outputs show semantically segmented images with additional separation between the different car instances. This car instance segmentation is an important initial step towards potential filtering of moving cars needed in later project stages. Despite the abundant number of cars in the considered test scenes, the model is able to realize an efficient distinction. Additionally, our model is be able to segment cars having similar colors and shapes from each other despite occlusion and superposition (see Fig. 2.18). It is as well obvious that the model is particularly aware of the aimed scenario. In fact, it is able to semantically segment all objects into semantic classes while performing instance-level segmentation just for cars.

Always in the goal to challenge the model robustness, we test it on random images selected from other datasets that it has never been acquainted with. The new datasets present the extreme scenarios with new environments and landscapes captured in other countries under different conditions with various cameras. Fig. 2.19 displays results on sample images from the extra data made available by Cityscapes. Although recorded in Germany, these images depict new cities that are not available in none of the training, testing and validation sets of the original setup. MokaNet indeed is just as performant on these images as with the original test set (Fig. 2.18). Similarly, it shows comparable visual quality with images recorded in USA from the BDD100K dataset [4] (see Fig. 2.20) and from



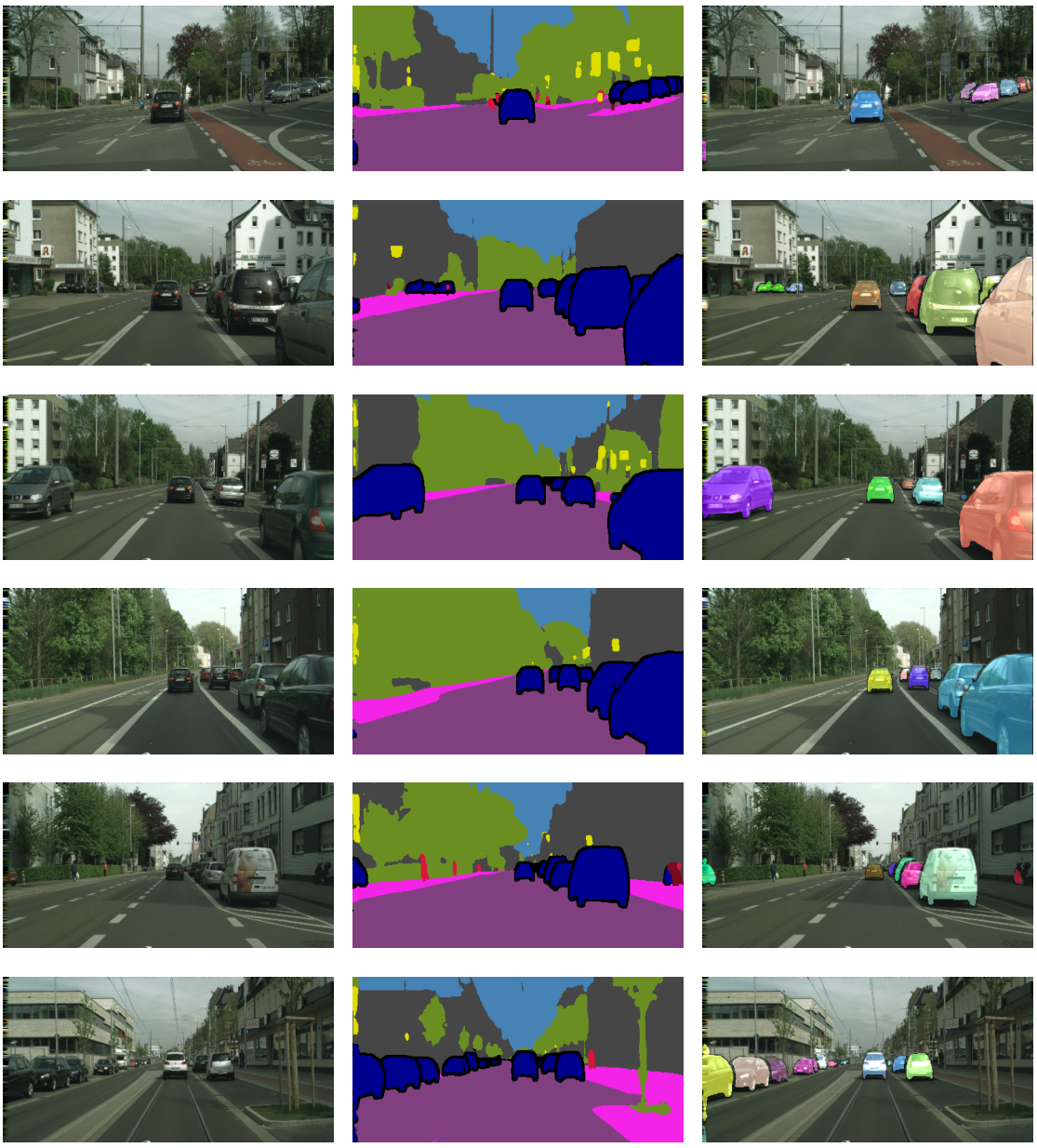
(a) Input frame (b) Estimated output mask (c) Input frame with blended car instances

Figure 2.18: Visual Evaluation on Cityscapes Test Set: (a) Example frames from test set (b) The estimated output mask with the car instances separately isolated (c) estimated transparent car instances blended onto the input frame.

our own recorded images in China (see Fig. 2.21) despite the significant environment and landscape differences. This highlights the robustness of the model trained only on German traffic images and gives a further confirmation about the efficient training procedure proposed including the heavy data augmentation. The comparison against best performing state-of-the-art methods from the Cityscapes benchmark with available public code shown in Tab. 2.8 confirms that MokaNet lands in second position after PolygonRNN++ introduced in [57] despite our much simpler structure.

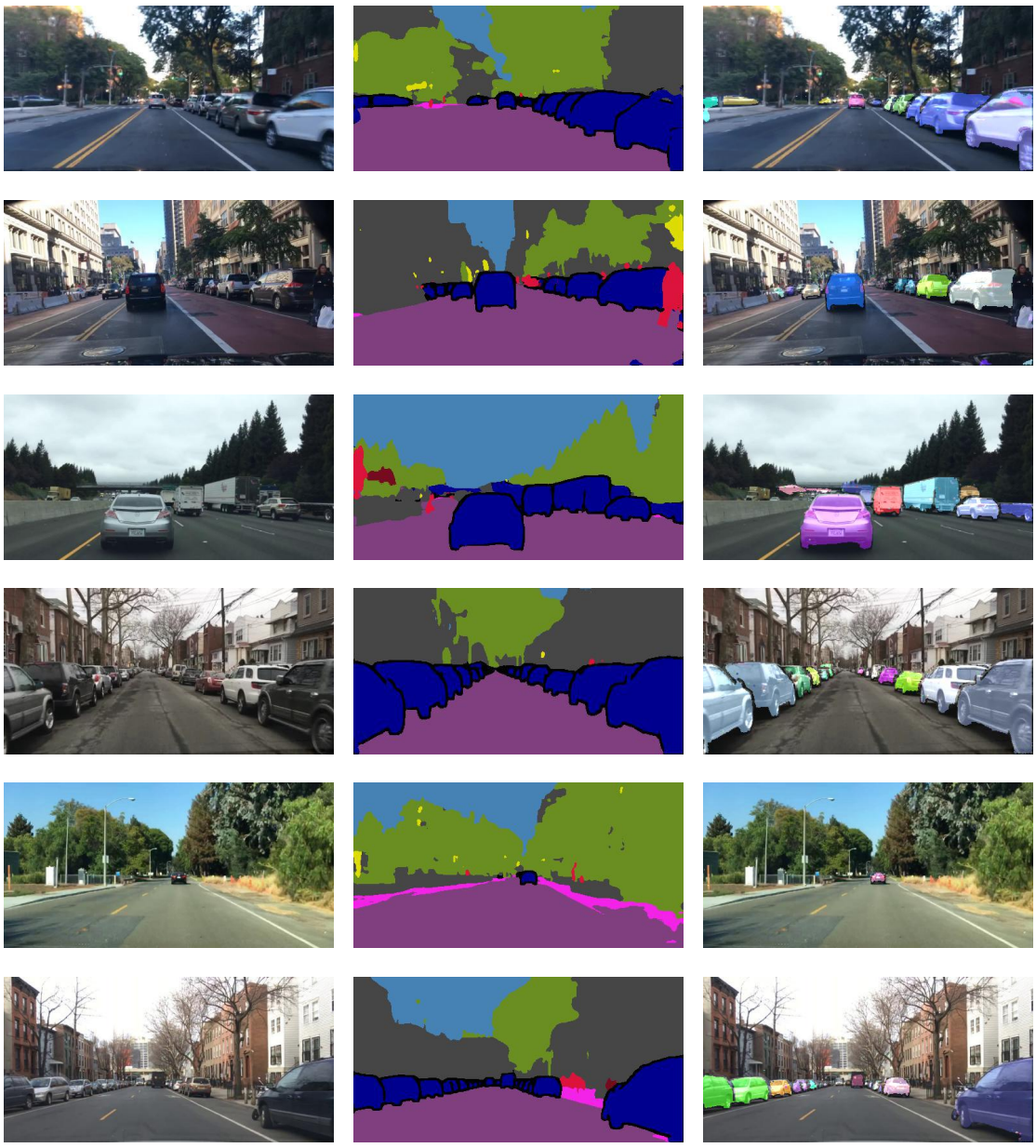
Method	Class	AP	$AP_{50\%}$
MokaNet	Cars	16.0	40.0
Multitask Learning	Cars	15.8	25.0
SIS with a Discriminative Loss Function	Cars	13.0	31.1
PolygonRNN++	Cars	25.4	45.0

Table 2.8: Car Instance Segmentation: The evaluation of our approach MokaNet against state-of-the-art on the validation set (500 image) of Cityscapes [1] with resolution (512×256).



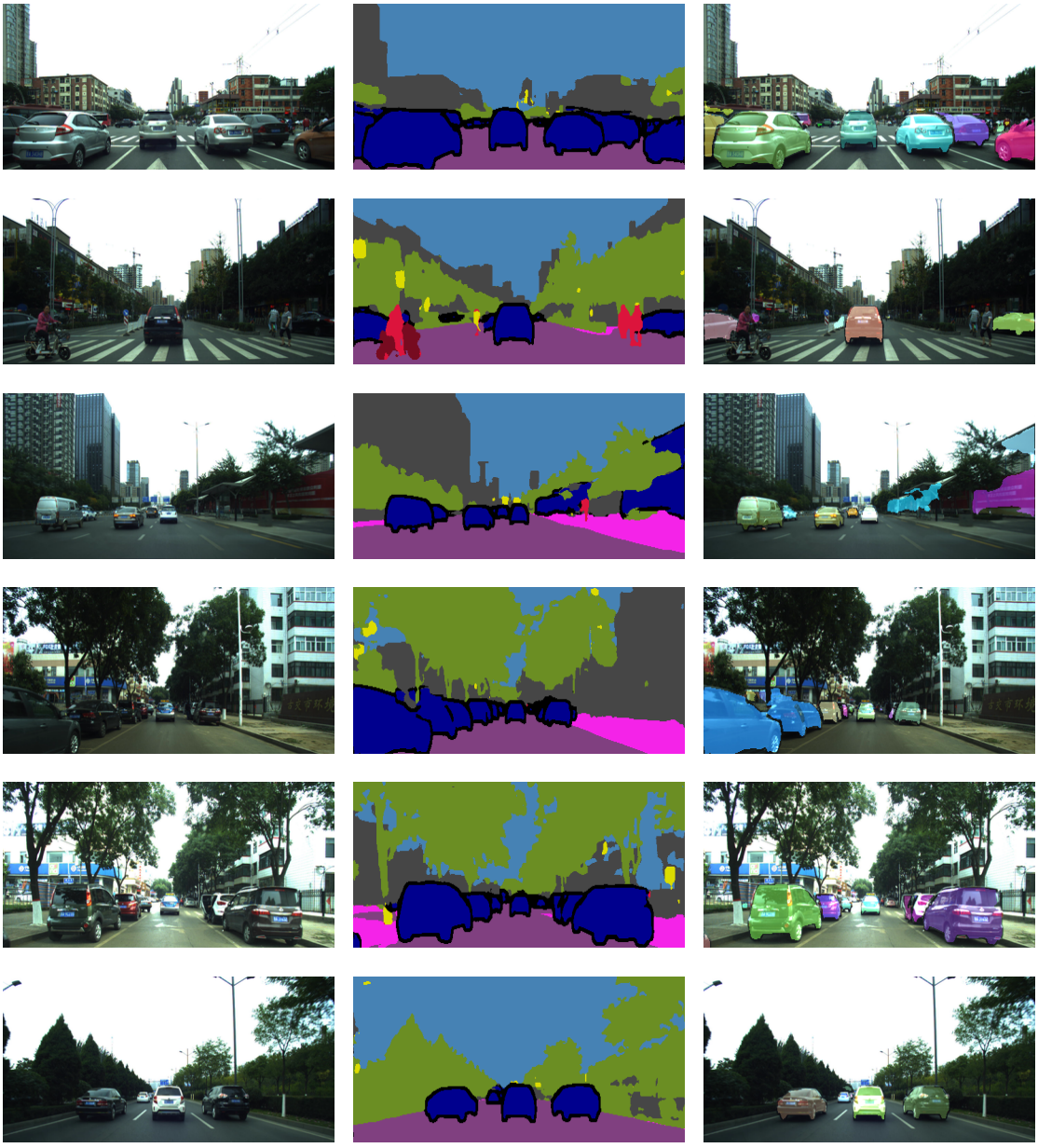
(a) Input frame (b) Estimated output mask (c) Input frame with blended car instances

Figure 2.19: Visual Evaluation on Germany Images: (a) Example frames from the "Trainextra" Cityscapes set (b) The estimated output mask with the car instances separately isolated (c) estimated transparent car instances blended onto the input frame.



(a) Input frame (b) Estimated output mask (c) Input frame with blended car instances

Figure 2.20: Visual Evaluation on USA Images: (a) Example frames from the BDD dataset [4] (b) The estimated output mask with the car instances separately isolated (c) estimated transparent car instances blended onto the input frame.



(a) Input frame (b) Estimated output mask (c) Input frame with blended car instances

Figure 2.21: Visual Evaluation on China Images: (a) Example frames from China recorded frames (b) The estimated output mask with the car instances separately isolated (c) estimated transparent car instances blended onto the input frame.

3 Multi-tasking With Supervised Applications: The Segmentation Effect

3.1 Motivation & Related Works

Focusing on optimizing a particular metric is generally enough to achieve reasonable performance, it is however likely that additional information could significantly improve the outcome. Extra information about related tasks to the one at hand can be of high contribution to accelerate and enhance the convergence. Shared representations between tightly related applications are likely to allow for better generalization of CNN models. The transfer of shared features between tasks introduces a kind of inductive bias that enlarges the hypothesis space. In fact, a model that performs well for a sufficiently large number of training tasks, is likely to perform well in learning novel ones coming from the same environment and drawn from the same distribution [58]. In DL, we refer to this concept as "multi-Task learning" or "multi-tasking". It revolves around the fact that a certain model is trained to perform a multitude of related tasks simultaneously such that they benefit from each other during training.
In the context of scene understanding, depth/disparity estimation and semantic segmentation seem to be perfect candidates for such approaches. More specifically, the growing interest in autonomous driving technologies made both tasks core research topics. Both are crucial for understanding and analyzing the actual environment surrounding the autonomous vehicle. The idea is then to explore the effect of combining the learning of semantic segmentation and depth estimation features. How beneficial could it be to jointly learn both tasks simultaneously? In particular, how does high quality segmentation affect depth estimation in such cases?
In reality, various research works tried to address similar questions. The recent work by Jafari *et al.* [59] proposes a modular network architecture that jointly solves depth estimation and semantic segmentation given a single image input. Their solution put a specific focus on the analysis of the the cross-modality's influence on the joint refinement of both tasks rather than quantifying the accuracy improvement. The proposed architecture fuses state-of-the art results for depth estimation [60] and semantic labeling [21] by properly balancing the cross-modality influences between both outputs. Wang *et al.* presented in [61] a unified model to jointly predict depth and segmentation. It produces fine-level detailed estimations by decomposing the input into local segments for region-level prediction od depth and segmentation under the guidance of a global layout. The final inference problem is formulated as a two-layer Hierarchical Conditional Random Field (HCRF) to produce the final depth

and semantic maps. The approach introduced in [62], addresses the task of joint inference of 3D scene structures and the semantic labeling of monocular videos by deriving a Conditional Random Field (CRF) model defined in the 3D space. The model is expected to jointly convey the semantic category and occupancy of voxels. More informed priors and constraints are then acquired, which is otherwise not the case if solved separately. Similarly, Mousavian *et al.* [63] show that combining depth and semantic segmentation cues can improve depth estimation and/or semantic segmentation performance in indoor scenarios. They propose an approach where semantic segmentation and depth estimation are jointly estimated from a single RGB input. Their architecture has two separate main modules. Each one is responsible for solving a single task corresponding to a pre-defined proper loss function. Moreover, Hazirbas *et al.* in [64] achieved interesting performance improvement by combining depth and segmentation processing in the encoder part despite the average depth quality produced by the Kinect. The FuseNet model, extracts features from RGB and depth and fuses them into RGB feature maps as the network goes deeper. This fusing proved indeed beneficial for the refinement quality of the segmentation task in particular. In the other hand, Zhang *et al.* tackled the problem of instance-level segmentation coupled with depth ordering from single image inputs. Using an end-to-end model, they produce instance-level image segments where the ID of each object encodes the depth ordering. Kendall *et al.* in [45] use a multi-tasking network able to solve semantic segmentation, depth estimation and instance level segmentation at once. Their model harness uncertainty-based weighting between loss terms corresponding to different tasks in the target of improving the understanding of autonomous driving scenes. In fact, they prove that combining these tasks ensures better agreement between the separate outputs while reducing computation at the same time. Following the same path, Uhrig *et al.* [44] presented an approach that leverages a classification model to predict semantic labels, depth and instance-based encoding using each pixel's direction towards the corresponding object center. Moreover, the solution in [60] addressed successfully three tasks with a unified model architecture. They are able to predict depth, surface normals, and semantic labeling. Yamagushi *et al.* combined even the solving of four tasks with their slanted plane model [65] for jointly recovering segmentation masks, dense depth estimates, flow predictions as well as boundary labels (such as occlusion boundaries) from a static scene given a stereo pair captured from a moving vehicle. The solution is an optimization algorithm for a SLIC-like objective which preserves connectedness of image segments and exploits shape regularization in the form of boundary length.

3.2 Multi-objective Optimization: Theory and Background

In order to evaluate the advantage of multi-tasking models in comparison to standard single ones, we need to take a look at the core design idea shared by most of the aforementioned works. Obviously, a CNN is an optimization tool. The set of parameters to optimize are the layer weights and biases and the target function is the pre-defined objective (cost, residual)

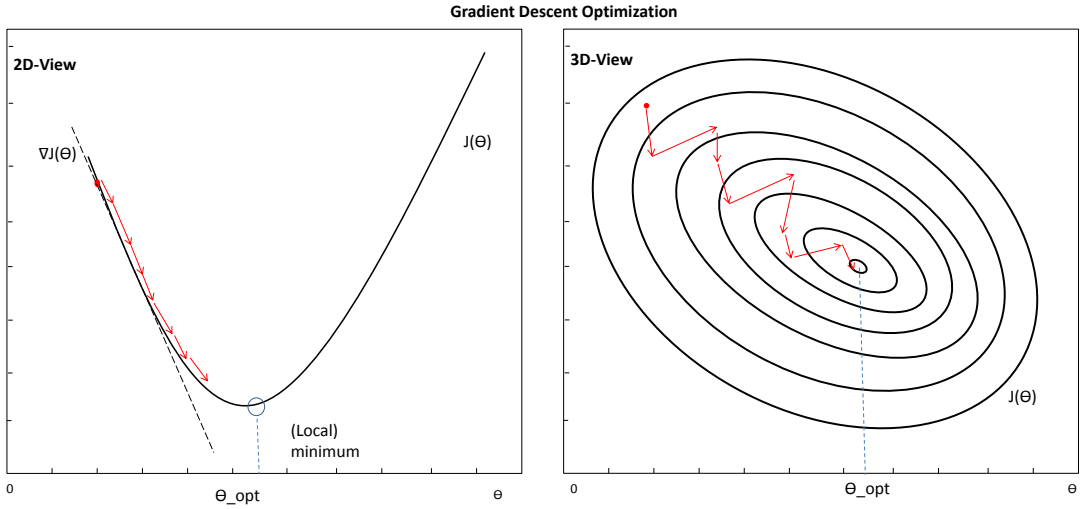


Figure 3.1: The gradient descent optimization strategy.

of the model.
For solving tasks using CNNs, we define the objective function $J(\theta)$ as the mapping we want to estimate from the training data. θ is the vector of learnable network parameters. The goal is to find the optimal θ (set of parameters) that minimizes the function $J(\theta)$ and generalizes the best over the available training set. In order to reach this, different gradient descent strategies can be adopted to keep updating the initial randomly initialized θ until the minimum of cost function $J(\theta)$ is found. If convexity is assumed, such procedure can be imagined as an iterative descent walk down a hill (simulating the function). The learning rate η defining the relative intensity of the weights update after each iteration is the step size we take each time on the way down towards the end (minimum). For each iteration, different directions stand open for taking a step. Hence, a proper direction choice must be taken to optimize the journey. The straight forward pick in our case is to keep always targeting the bottom of the hill. This is the role of the term $-\nabla_\theta J(\theta_+)$ that describes the steepest descent direction at a certain point θ_+. To sum up, we can formulate the minimization problem as:

$$\theta_+ = \arg\min_\theta J(\theta) \qquad \text{s.t} \quad \theta \in R^n \tag{3.1}$$

A new point θ_+^{k+1} is computed after every step k until the optimum is reached using the update formula:

$$\theta_+^{k+1} = \theta_+^k - \eta \cdot \nabla_\theta J(\theta_+) \tag{3.2}$$

Typically, the objective function $J(\theta)$ defines the average loss L over a given training set TrD. The mathematical description of a certain task can be written as $f(x, \theta)$ that is

dependent on the given input x and network parameters θ at a specific iteration. Additional terms independent from θ are potentially inherent in L as well. These can be the ground-truth value y in the case of supervised learning or an other θ-free term in case of unsupervised applications. For instance, if we consider the optimization of our semantic segmentation problem, the corresponding objective function could be written as:

$$J(\theta) = E_{TrD}\left\{\ L(f(x,\theta),y)\ \right\} = L_1(\theta) \tag{3.3}$$

where L_1 is the loss term corresponding to the optimization of the semantic segmentation.

However, optimization problems involving more than one single task could be differently formulated. In reality, there are different alternative ways to approach this.
Such type of problems is generally referred to as multi-objective optimization also known as multi-criteria, Pareto, multi-attribute, multi-performance or simply vector optimization. It is a mathematical problem addressing the simultaneous optimization of more than one objective function. In other words, it is a tool for treating complex systems by concurrently optimizing several criteria at once. The basic formulation is:

$$\arg\min_{\theta} J(\theta)^T = \arg\min_{\theta} \left(L_1(\theta), L_2(\theta), ..., L_n(\theta)\right) \tag{3.4}$$

In fact, the objective function to be minimized is not a scalar anymore but of vector form and can be written as:

$$J(\theta) = \begin{pmatrix} L_1(\theta) \\ L_2(\theta) \\ \cdots \\ L_n(\theta) \end{pmatrix} \tag{3.5}$$

Understandably, there is not necessarily a single solution that simultaneously optimizes each objective but rather a set of different possible trade-offs. In the case of functions with different minima, a solution is reached when no objective function can be optimized without degrading the others. The solution establishes then an trade-off between the different objectives and it is called efficient or pareto-optimal. Finding the pareto optimal set with all possible efficient solutions depends tightly on the formulation and the initial set up of the optimization problem. Several methods have been suggested to solve such complex optimizations.
According to multi-objective optimization literature e.g. [66] [67]..., the most intuitive approach to address such problems is the classical method based on function scalarization. For this, a vector function is transformed into a scalar one involving the weighted single objectives combined as a global criterion.
let us assume we have n tasks with their corresponding loss terms $L_1, L_2 \cdots L_n$ respectively. In order to optimize all functions simultaneously using the scalarization method without prioritizing any of the tasks, we can reformulate the global objective function $J(\theta)$ describing the new problem as a weighted summation of all loss terms:

$$J(\theta) = \sum_{i=1}^{n} \alpha_i\, L_i(\theta) = \ \alpha_1\, L_1(\theta) + \cdots + \alpha_n\, L_n(\theta) \tag{3.6}$$

The resulting joint function is the substitute model for the vector optimization problem described by equation. 3.5. It consists of agglomerating the different independent criteria $L_i(\theta)$ into a single convex one. Appropriate weights $\alpha_i \geq 0, \forall i = 1, ..., n$ are assigned to each single term in order to combine them in a weighted sum. If all tasks are equally prioritized, then the weighting factors shall be equal $\alpha_1 = \alpha_2 = ... = \alpha_n$. In the case of a higher prioritized task, a higher factor is assigned to this particular one. Ultimately, the reformulated problem can be approached as a single scalar objective function. To solve it, standard optimization can be performed. The reformulated problem is defined as:

$$\arg\min_{\theta} \sum_{i=1}^{n} \alpha_i \, L_i(\theta) \tag{3.7}$$

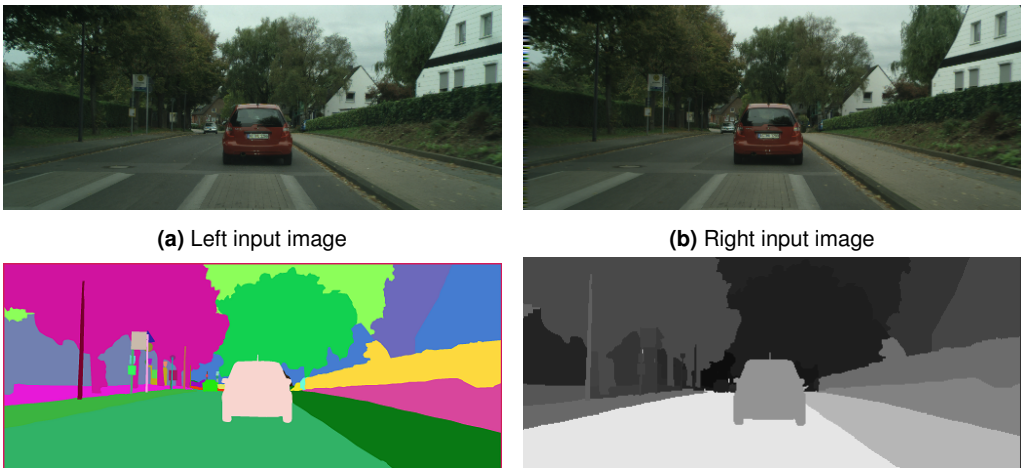
As we restrict ourselves to the case of equally prioritized tasks, the corresponding loss terms of the different tasks are always equally weighted all along this research work. In other words, no preference is accorded to semantic segmentation in respect with other applications when performing multi-tasking.

3.3 Cooperative Optimization For Disparity Planes Estimation And Semantic Segmentation

3.3.1 Disparity Planes Estimation

Method Description

The required project specifications previously discussed offer the possibility to consider many important additional image cues for improving urban scene understanding applied on autonomous driving applications. In the previous chapter, we discussed the semantic segmentation on class and instance-levels. With the stereo-recording setup that the project requirements specified, interesting perspectives can be additionally considered.
Obviously, the most important aspect about stereo-vision is the triangulation concept. This revolves around determining points in 3D space given their projections onto two or more images. The absolute three-dimensional location of a point is also called *depth*. Once projected under different perspectives, this information can be estimated considering the difference between the locations of the projections. This relative difference called *disparity* is proportional to the 3D real world information (depth). Therefore in the context of stereo-vision, referring to one of the concepts is necessarily referring to the other as they are inversely proportional. However, the question that persists in our regard is, how is it possible to incorporate this additional source of information together with semantic segmentation in order to improve scene understanding? More specifically, how can the estimation of depth/disparity benefit from high quality semantic segmentation?
In stereo vision, the perception of depth arises from the disparity of a given 3D point which



(a) Left input image

(b) Right input image

(c) Input segmentation mask for left image

(d) Output: disparity planes ground-truth label

Figure 3.2: The ground-truth generation process for disparity planes estimation (left view)

is again the distance between the point projections onto the two images when superimposed. In general, the disparity Z and depth D relates to each other as:

$$Z = \frac{fB}{D} \tag{3.8}$$

with f focal-length and B the baseline are two pre-fixed constants.
Underpinned by this proportionality, we can rely on the disparity information that we gain from the stereo-vision setup to have an idea about the 3D world. After exploring the semantic segmentation aspect, the first step to combine this with the disparity cue is to consider the relative distance between segments corresponding to the same object in two stereo images. This task we call it disparity planes estimation as it considers each object to have a constant average disparity value and therefore lying in a single independent depth plane.

In order to estimate disparity planes, a stereo image pair is required along with the segmentation mask for one of them. Each single segment in the given mask is expected to be found in the opposite image. Once this is done, the displacement can be directly estimated which subsequently means that the corresponding disparity plane is defined.
Let us assume we have a left image I_l and a right one I_r that are given. We assume also that the semantic segmentation mask S_l for the left image is given with N segments s_i, $i = 1..N$ depicting differently labeled semantic classes. We can define s_i as the set of coordinate pairs for pixels belonging to the same class i. Assuming that the range of disparity values is pre-known $D = \{0,...,d_{M-1}\}$ with M possible disparity levels, we consider a segment $s_i = \{(x_i^k, y_i^k) \quad | \quad S_l(x_i^k, y_i^k) = i, \quad k \in \{1..K\}\}$ with K pixels belonging to the same class and having corresponding coordinate pairs (x_i^k, y_i^k). We calculate the corresponding mapped set:

$$s_i^d = \{(x_i^k + d, y_i^k) \quad | \quad k \in \{1..K\}, \quad d \in D\}$$

This corresponds to the sliding of the segment points along the x-axis using a single disparity value $d \in D$. We calculate the resulting photometric error between pixels with coordinates (x_i^k, y_i^k) belonging to the region defined by s_i in the left image I_l and pixels defined by s_i^d in the right image I^r as:

$$e_i^k(d) = ||I_l(x_i^k, y_i^k) - I_r(x_i^k + d, y_i^k)|| \quad | \quad d \in D, k \in \{1..K\}, i \in \{1..N\} \tag{3.9}$$

The averaged photometric error over the whole segment s_i is defined as:

$$e_i^{aver}(d) = \frac{1}{K} \sum_{k=1}^{K} e_i^k \quad | \quad d \in D, i \in \{1..N\} \tag{3.10}$$

The best disparity value d producing the minimal average error $e_i^{aver}(d)$ is considered as the relative constant disparity for this segment (see example in Fig. 3.2).
Going through all the segments within the image, the produced output should be a kind of disparity map where each segment has a single constant disparity value defining its relative displacement between the two images. That is why, we can consider the approach as a sort of disparity ordering/layering algorithm. The advantage of the generated output is that it first sustains semantic separation between the objects. Second, the information encoded by the ID-labels (single disparity value) for each segment gives an approximate estimate of the real-world distance of the scene objects.

Training Setup

Referring to the described procedure, we generate the set of ground-truth labels of all Cityscapes training images for the task of disparity planes estimation. We harness the instance-level ground-truth available for each stereo pair. We apply the previous procedure to generate the new labels for all left views. The final resulting dataset consists of 2975 input pairs for training and 500 pair for validation. All pairs get their corresponding disparity planes ground-truth labels for left views generated. For testing, no instance ground-truth information is available. That is why, disparity planes ground-truth cannot be generated.
Similarly to experiments in chapter 2, we opt for the same data augmentation strategy in order to enrich our training set by adding 4 additional color-mapped stereo pairs for each original one. For training, the final set contains 11900 pairs and the same online augmentations previously used are applied. The only difference is that consistency between image pairs should be maintained. In fact, the same set of transformations is applied each time on both input images without difference.
After creating the dataset, we design the CNN for the new application. Obviously, we opt for the same cascaded MokaNet architecture (Fig. 2.3). In this case, the input is a stereo pair and the output is the estimated disparity planes mask. We consider the same setup for training, namely Caffe [51] with Adam optimizer [52] where $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

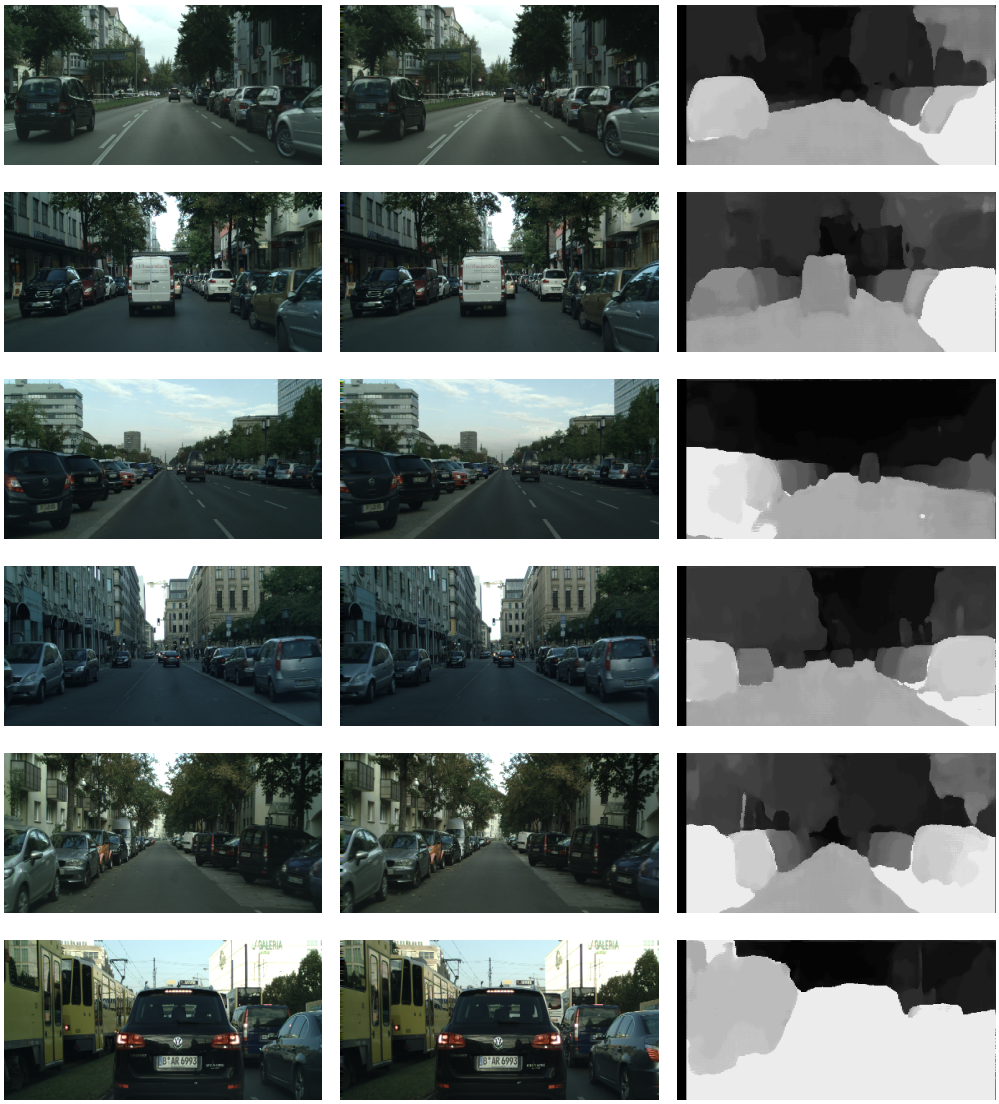
Batch size is 2 and maximum iteration is $3.5M$. The starting learning rate $\lambda = 10^{-4}$ is progressively updated by a factor of 0.5 every $150K$ iterations. The used decay factor and the gamma value are again $\delta = 10^{-4}$ and $\gamma = 0.5$ respectively.

Results

After training, the final model is deployed on test images of Cityscapes. Visual results are shown in Fig. 3.3 with the input stereo pairs and their estimated disparity planes outputs. Numerical results on the validation set are depicted in Tab. 3.1. We consider several metrics from prior works in the context of depth/disparity estimation in order to evaluate the model performance.
let us assume the evaluation of a certain image I against its ground-truth $\bar{I}$ (with N total number of pixels, x_i and y_i are coordinates along x and y-axis respectively). The used metrics could be summarized as:

- Abs Relative difference:
$$\frac{1}{N}\sum_{x_i,y_i}\frac{|I(x_i,y_i)-\bar{I}(x_i,y_i)|}{\bar{I}(x_i,y_i)}$$
- Squared Relative difference:
$$\frac{1}{N}\sum_{x_i,y_i}\frac{||I(x_i,y_i)-\bar{I}(x_i,y_i)||^2}{\bar{I}(x_i,y_i)}$$
- RMSE:
$$\sqrt{\frac{1}{N}\sum_{x_i,y_i}||I(x_i,y_i)-\bar{I}(x_i,y_i)||^2}$$
- RMSE (log variant):
$$\sqrt{\frac{1}{N}\sum_{x_i,y_i}||log(I(x_i,y_i))-log(\bar{I}(x_i,y_i))||^2}$$
- Mean Log_{10}:
$$\frac{1}{N}\sum_{x_i,y_i}|log_{10}(I(x_i,y_i))-log_{10}(\bar{I}(x_i,y_i))|$$
- Threshold: % of pixels s.t $max(\frac{I(x_i,y_i)}{\bar{I}(x_i,y_i)},\frac{\bar{I}(x_i,y_i)}{I(x_i,y_i)}) = \delta \leq$ thr

Visual and numerical results confirm that our final model is able to approximate the mapping between input (left and right images) and ground-truth (disparity planes labels) despite its complexity. However, results show that estimating disparity planes is more challenging than learning semantic segmentation. Despite using the same training setup,



(a) Left input frame

(b) Right input frame

(c) Estimated output disparity planes mask for left view

Figure 3.3: Visual Evaluation on Cityscapes Test Set: (a)&(b) Example stereo frames from test set (c) The estimated output mask for disparity planes

visual comparison of output quality shows that the same architecture performs better in semantic segmentation than disparity planes estimation. The quality difference and the relative poor performance proven visually and numerically for this experiment can be explained by the complexity of the mapping at hand compared to the previous experiment. In fact, this does not only consists of segmenting objects but also requires a geometrical approximation for estimating their displacements between left and right views.
Moreover, the quality of the used ground-truth labels can be a weak point for this experiment. The nature of the ground-truth generated is indeed approximate. The previously detailed procedure for generating disparity planes masks has been conceived as an experimental trial to examine the behavior of CNN models with such highly complex tasks. More specifically, the motivation behind the experiment was initially to combine segmentation and stereo-related cues without focusing on the ground-truth accuracy. We mainly aimed to prepare an experimental setup to test the effect of combining the semantic segmentation with stereo-related tasks and explore the full potential of multi-tasking when proper semantic segmentation is available. For this, the current results of our experiment should obviously fulfill this role regardless the quality.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Disparity Planes (DP)	0.0314	0.6350	9.3132	00.5377	0.1056	0.6386	0.8450	0.8891

Lower is better
Higher is better

Table 3.1: Numerical evaluation of disparity planes estimation on Cityscapes validation set with resolution (512×256)

3.3.2 Disparity Planes Estimation & Semantic Segmentation

Introduction

In order to examine the effect of segmentation on the disparity planes estimation, we design a complex multi-tasking architecture for learning both tasks simultaneously. First, this model contribute to the improvement of scene understanding as it delivers not only semantic segmentation outputs but also an estimate of the disparity ordering of the scene. Besides, it allows for an evaluation of cooperative multi-objective optimization against standard single methods. Finally, it gives us insight on the effect of associating semantic segmentation to a supervised stereo-related application within a multi-tasking framework. In this experiment, the assumption that high quality semantic features are likely to boost learning stereo cues is to be examined and verified based on quantitative and qualitative evaluation.

Proposed Approach

In order to examine the merits of joint disparity planes estimation and semantic segmentation, we design a customized architecture inspired by the model previously used in our experiments (MokaNet). However, the new model we propose in Fig. 3.4 has a bifurcation at the end of the encoder where two decoders instead of a single one are connected. Each one is responsible for decoding features corresponding to a single application. The estimated output of the first track is the disparity planes estimate while the second track is for the semantic segmentation. The model fuses information coming from both tasks all along the encoding part but separates these during the recovery phase so that every loss term is optimized independently.

The motivation behind the described design is that shared encoder features carry more

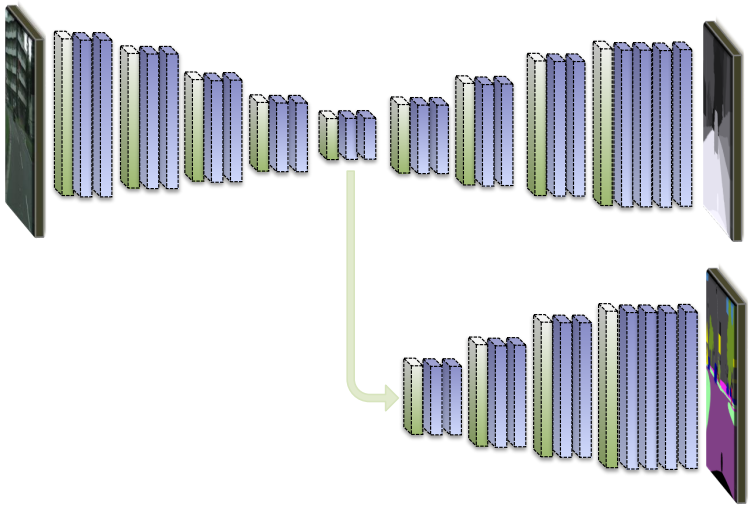


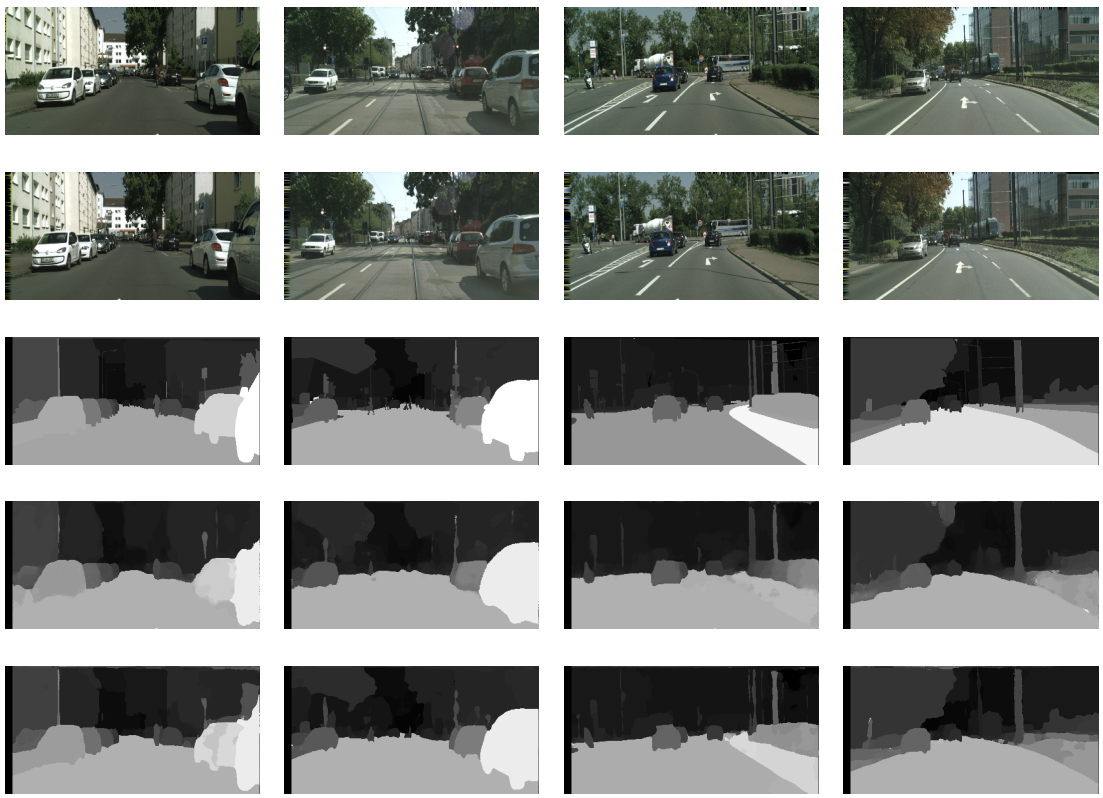
Figure 3.4: The proposed architecture for cooperative optimization (multi-tasking CNN): Joint disparity planes estimation and semantic segmentation

information and details in comparison to features learned for a single application. By considering two tasks, the learning process has access to a larger pool of information. Mathematically, the back-propagation step responsible for the update of the network's weights propagates gradients from both tracks in the reverse way. Hence, the encoder shared weights undergo the effect of both optimization problems at the same time during every back-propagation step. If both tasks are tightly related, richer feature maps with better encoded information at the end of the constructive part are gained. If this is the case, the decoding part will experience as well a quality improvement in terms of recovered details. For training, we choose the same configuration including the data augmentation strategy

(4 additional color-mapped stereo pairs for each original input pair resulting in set of 11900 pairs additionally to the data augmentation applied online), and the same setup with Caffe [51] as framework and Adam as optimizer [52] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, batch size of 2, maximum iterations up to $3.5M$, $\lambda = 10^{-4}$ multiplied by a factor of 0.5 every $150K$ iterations, $\delta = 10^{-4}$ and $\gamma = 0.5$.

Results & Discussion

A visual assessment of the network's performance for estimating the disparity planes on the validation images of the Cityscapes dataset is presented in Fig. 3.5. The visual comparison against the results of the single optimization problem from the previous experiment and the available ground-truth labels proves that the multi-tasking model delivers slight improved results in terms of quality. This observation buttresses the assumption that the disparity planes estimation profited from semantic segmentation features learned simultaneously. Numerical results presented in Tab. 3.2 confirm the fact that segmentation features helped decrease the noise and refine the shapes of different scene objects. They also enforced better separation between the segments with a noticeable improvement of car instance distinction. Obviously, the semantic segmentation track that has this information encoded in its ground-truth has helped highlighting these aspects. Numerical values show a low but consistent improvement of the multi-tasking model upon the basic model considering all measures. This can be as well verified in Fig. 3.6 where visual comparison between results of both models are examined on the test set of Cityscapes.
Understandably, previous observations concerning the performance of the single model are still applicable on this experiment as well (Mapping complexity and the ground-truth reliability issues). However, the comparison delivers an undeniable confirmation about the assumption made concerning an improved learning of segment displacements when semantic segmentation features are considered at the same time during the training. The depicted object shapes, edges and details along the contours experienced a significant improvement with less inherent noise.
After verifying the positive effect of semantic segmentation on disparity planes estimation. we extend the exploration from the segment level to the pixel level. The challenging case of pixel-wise displacement estimation (in other words standard disparity estimation) is targeted. The goal is to examine if the improvement gained thanks to additional segmentation learning can be also valid with more complex geometrical tasks than disparity planes estimation. How does semantic segmentation affect the task of supervised stereo disparity/depth estimation when they are jointly learned?



(a) 1.Scene (b) 2.Scene (c) 3.Scene (d) 4.Scene

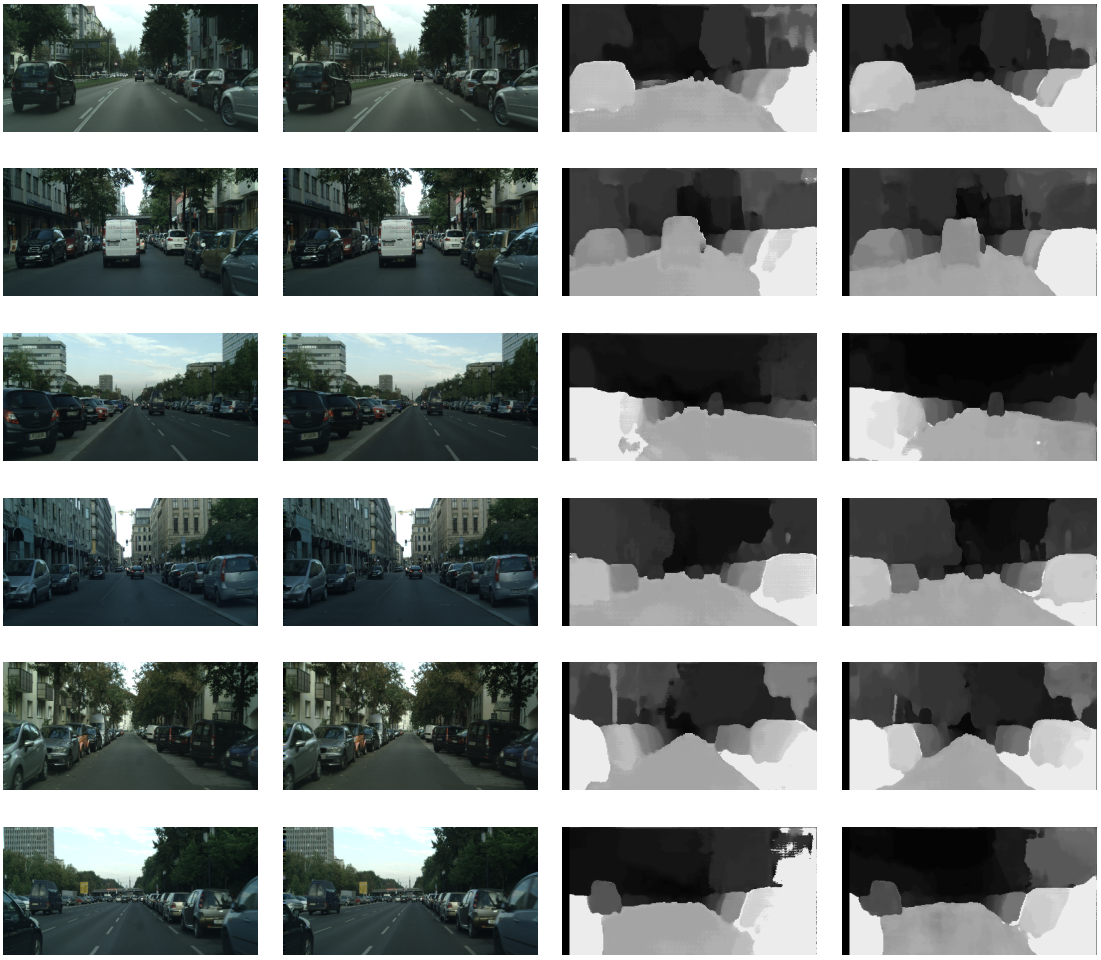
Figure 3.5: Visual Evaluation on Cityscapes Validation Set. From top: Left image, right image, ground-truth disparity planes for left view, disparity planes estimated output for left view (single model), disparity planes estimated output for left view (multi-tasking model).

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Single model	0.0314	0.6350	9.3132	0.5377	0.1056	0.6386	0.8450	0.8891
Multi-tasking model	0.0244	0.5350	9.1349	0.4683	0.0870	0.7182	0.8847	0.9122

Lower is better

Higher is better

Table 3.2: Numerical performance comparison on Cityscapes Validation set with resolution (512×256) for single and multi-tasking models



(a) Left input image

(b) Right input image

(c) disparity planes estimated output for left view (single model)

(d) disparity planes estimated output for left view (multi-tasking model)

Figure 3.6: Visual Evaluation on Cityscapes Test Set: The visual comparison of models outputs in case of cooperative optimization (multi-tasking model) and single objective optimization (single model).

3.4 Cooperative Optimization For Supervised Depth Estimation And Semantic Segmentation

3.4.1 Supervised Depth Estimation: DispNet

Introduction

Although depth estimation is most commonly related to stereo vision (stereopsis), many methods try also to address the problem of monocular depth estimation. For instance, Saxena *et al.* [68] apply a Markov Random Field (MRF) learning algorithm to capture certain monocular cues and incorporate them into a stereo system. They show that by adding these cues to stereo (triangulation), performance got improved. Liu *et al.* [69] explored the combined capacity of CNN models and continuous Conditional Random Fields (CRF). Specifically, they propose a structure that learns the unary and pairwise potentials of continuous CRF in a unified deep CNN framework.
Nevertheless, stereo methods have always enjoyed more attention and proved indeed more effective in the context of supervised learning in particular. For instance, works from Zbontar *et al.* [70] as well as Luo *et al.* [71] used rectified image pairs to extract depth information either by learning a similarity measure on small patches using CNNs or by treating the problem as multi-class classification using a Siamese network. The state-of-the art supervised approach is still the model called DispNet introduced by Mayer *et al.* in [72]. DispNet is inspired by FlowNet [49]. It is indeed a modified FlowNet version with additional new features for performing disparity estimation. The model is trained first on a large synthetic stereo dataset then fine-tuned on real-world images from KITTI [11].

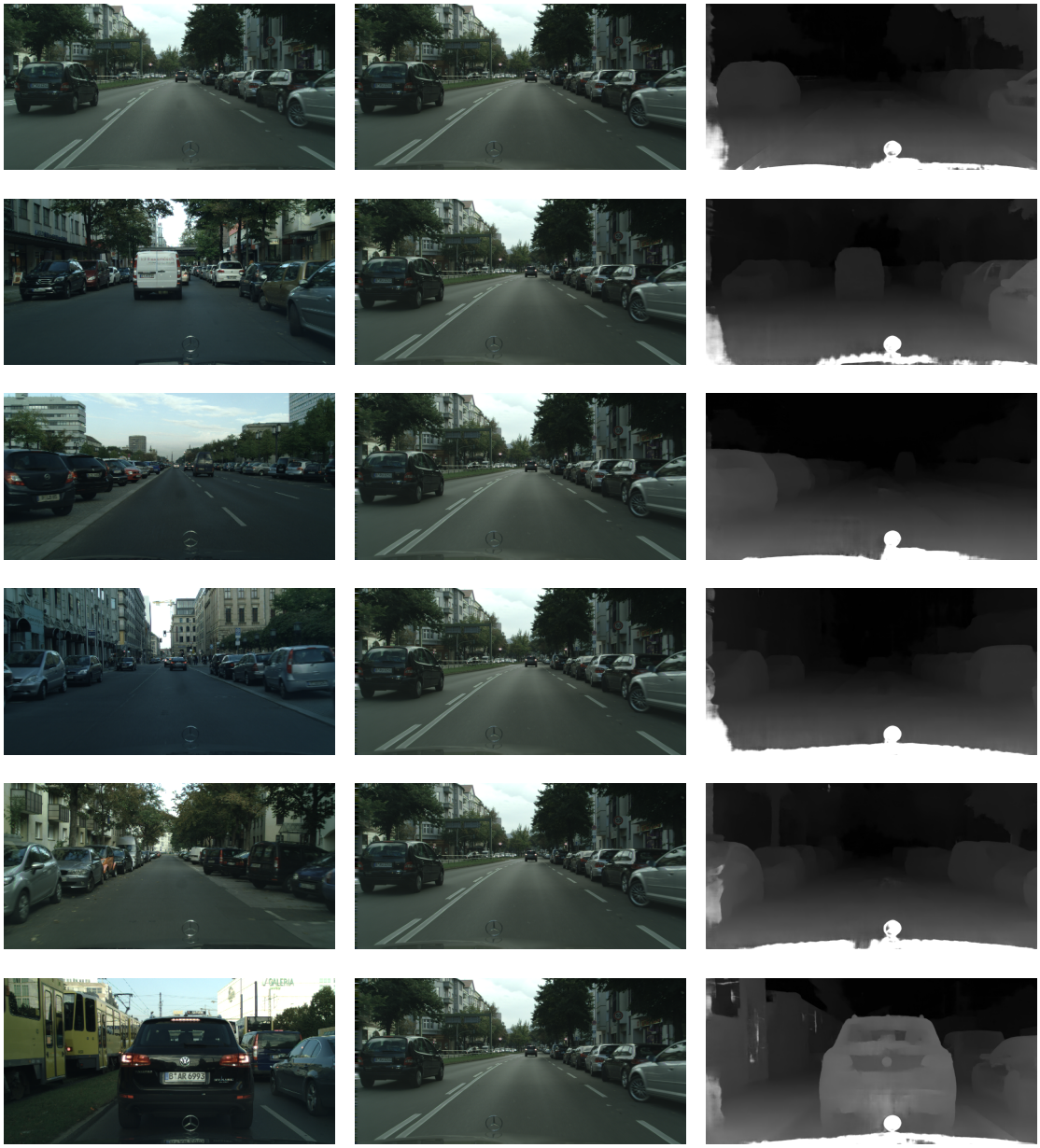
Implementation Details

As detailed in [72], DispNet is a CNN for disparity estimation given stereo images. It is trained on a synthetically generated dataset with considerable realism, variation and number of frames. Currently, it provides the best state-of-the-art performance for supervised disparity estimation. Therefore, we present in the following a brief description of the model. From the design point of view, DispNet embraces the original FlowNet architecture introduced in [49]. It has an encoder-decoder layout with few extra-links between the two parts. Unlike our proposed model, these are exclusively joining layers from both parts that output features with the same dimensions. Obviously, the linking strategy of DispNet is considerably lighter and less sophisticated in comparison to ours. The encoder contains a succession of five convolution layers with stride 2 resulting in a total down-sampling factor of 64 at the end of the encoding part. The decoder is symmetrically designed to up-sample the feature maps by taking into consideration only same-scale features linked from the encoder. For more details about the different DispNet versions, we refer to Dosovitskiy *et al.* [49].
Since getting a reliable depth/disparity ground-truth for real stereo images is a very te-

dious task, DispNet is first trained on a synthetically produced data collection consisting of the three sets: "Monkaa", "Driving" and "FlyingThings3D". The contained stereo frames depict various scenes that were rendered using "Blender" [73], an open source framework for creating, animating and rendering 3D objects with complex motions. These input images come with high quality synthetic disparity maps for each view. These maps rendered also using Blender are considered as ground-truth labels when training the original DispNet model (For further details concerning the dataset generation procedure refer to [72]). Different data augmentations are applied on the rendered dataset in order to bring more robustness to the model: Spatial transformations such as rotation, translation, cropping, scaling are performed along with photometric transformations on color, contrast and brightness in order to enforce more diversity within the synthetic training set. Regarding the training, it is realized using a custom version of Caffe [51]. Adam [52] is chosen as optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is $\lambda = 0.0001$ and divided by 2 every $200K$ starting from $400K$.

Results

A visual evaluation of the DispNet performance on test images of the Cityscapes dataset is shown in Fig. 3.7. Although trained on synthetic data only, the model proves effective in estimating depth for real stereo images depicting traffic urban scenes. It is indeed able to generalize properly thanks to the abundant set of data used for training and the different data augmentation operations applied on it. DispNet is able to robustly estimate the displacement on pixel level without being strongly affected by the real environment in the test scenes or the content variation in comparison to the training images.



(a) Left input frame (b) Right input frame (c) Estimated output: Left depth map

Figure 3.7: Visual Evaluation on Cityscapes Test Set: (a) & (b) Sample input stereo pairs from test set (c) The estimated output left depth map

3.4.2 Supervised Depth Estimation & Semantic Segmentation

Introduction

Proving the improvement gained by jointly optimized applications requires a complex architecture where features learned through separate tracks are fused or brought together in a way that boosts the training process. Like the experiments for disparity planes estimation, we want to verify that associating segmentation learning to supervised depth estimation in the context of autonomous driving helps improve the quality of the gained depth maps. In the previous experiments, disparity planes ground-truth labels were generated based on already available segmentation ground-truth made public by Cityscapes. Obviously, both ground-truth labels were accessible at the same time for each input pair. That is why no problem performing multi-tasking training was faced. Unfortunately, this cannot be replicated for the current experiment.
To our knowledge, there is no available dataset with real images for autonomous driving offering reliable ground-truth labels for depth estimation and semantic segmentation at the same time. Therefore, we restrict ourselves to train on the synthetically generated dataset (used for original DispNet training) since it has also ground-truth labels for object segmentation. These are rendered by Blender as classification masks. For deployment, the most known autonomous driving datasets previously used will be regarded for evaluation and assessment, namely the Cityscapes and KITTI datasets.
To sum up, a cooperative multi-tasking model is designed. It was trained on synthetic data then tested and evaluated on real images in the target to prove that the simultaneous learning of segmentation features along with supervised depth cues can result in significant improvement in terms of quality for the estimated stereo depth maps. For this, our proposed model is a combined architecture consisting of two tracks. The first is the state-of-the-art architecture for supervised stereo depth estimation (DispNet) and the second is our cascaded architecture previously introduced for semantic segmentation (MokaNet). The new model designed for the current experiment combines the learning of two tasks by enforcing feature exchange between these tracks during the training. It is evaluated against the original DispNet model to discuss the advantage of multi-tasking upon standard supervised depth estimation.

Implementation Details

As mentioned above, the proposed architecture for our new model that jointly performs semantic segmentation and supervised stereo depth estimation consists of a combination of the original DispNet architecture and our MokaNet proposed for semantic segmentation. Both tracks share a single encoder with five convolution layers as in the original DispNet architecture. At the level of the last encoding layer, the model presents a split into two independent decoders. The first one that is exactly the original DispNet decoder delivers the estimated disparity map as output. This output is post-processed onto depth based on the focal-length and baseline given by the test dataset. The second decoder that is our

intensively connected one is responsible for semantically segmenting the image. It has dense connections between layers and is borrowed from the architecture used previously for semantic segmentation. Like in the case of disparity planes estimation, the encoded information is shared between both tracks whereas the details recovery is independently done for each task thanks to a separate decoder.
As in the previous experiment, the fact that both objective functions simultaneously contribute to the updating of the encoder parameters (weights and biases) is expected to enhance the learnable features at the output of each track, the one for depth estimation in particular. With details coming from the segmentation learning process, edges, object shapes and forms are expected to be better depicted in the final depth maps compared to original DispNet. Although this assumption proved valid in the case of disparity planes estimation (the quality of the produced estimates by the multi-tasking model showed significant improvement in comparison to the single one), this needs to be properly examined for the case of supervised depth estimation as many differences are noted. The training data is indeed not only the same but it presents totally different signal nature. In the previous experiment for disparity planes estimation, we trained on real images from the training set of Cityscapes and tested on validation images of the same dataset. In the current experiment, we train on synthetic images and evaluate on real images from other different datasets which represents a significant difference against the previous experiment. Used architectures for the multi-tasking models are as well slightly different in terms of linking density which represents a further difference to be minded. For training, the exact same parameter setup used for the original DispNet is considered for training our new multi-tasking model in the goal to guarantee fairness and consistency of the final depth evaluation.
As detailed in [72], the original training set consists of the synthetically produced data collection including three parts "Monkaa", "Driving" and "FlyingThings3D" rendered with Blender. The considered frames have disparity maps and object segmentation masks for both views rendered also by Blender. They are used as ground-truth labels for training the multi-tasking model. (For further details on the training data and its generation procedure please refer to [72]).
We adopt as well the same training environment suggested by the authors. We employ a custom version of Caffe [51] with Adam optimizer [52] where $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\lambda = 0.0001$ divided by 2 every $200K$ starting from $400K$. We add the same set of augmentations applied by them: Geometric operations such as rotation, translation, cropping, scaling together with photometric transformations on color, contrast and brightness.

Results

In Fig. 3.8, a visual comparison on the test set of Cityscapes between the original DispNet and our cooperative multi-tasking model is shown. The estimated output depth maps produced by both models are displayed. As a result, the original DispNet model seems better performing than ours trained for depth estimation and segmentation simultaneously.

(a) Left input frame

(b) Right input frame

(c) Estimated output left depth map (single DispNet)

(d) Estimated output left depth map (multi-tasking model)

Figure 3.8: Visual Evaluation on Cityscapes Test Set: (a) & (b) Sample input frames from the test set and (c) & (d) are the estimated output depth maps using single and multi-tasking models respectively.

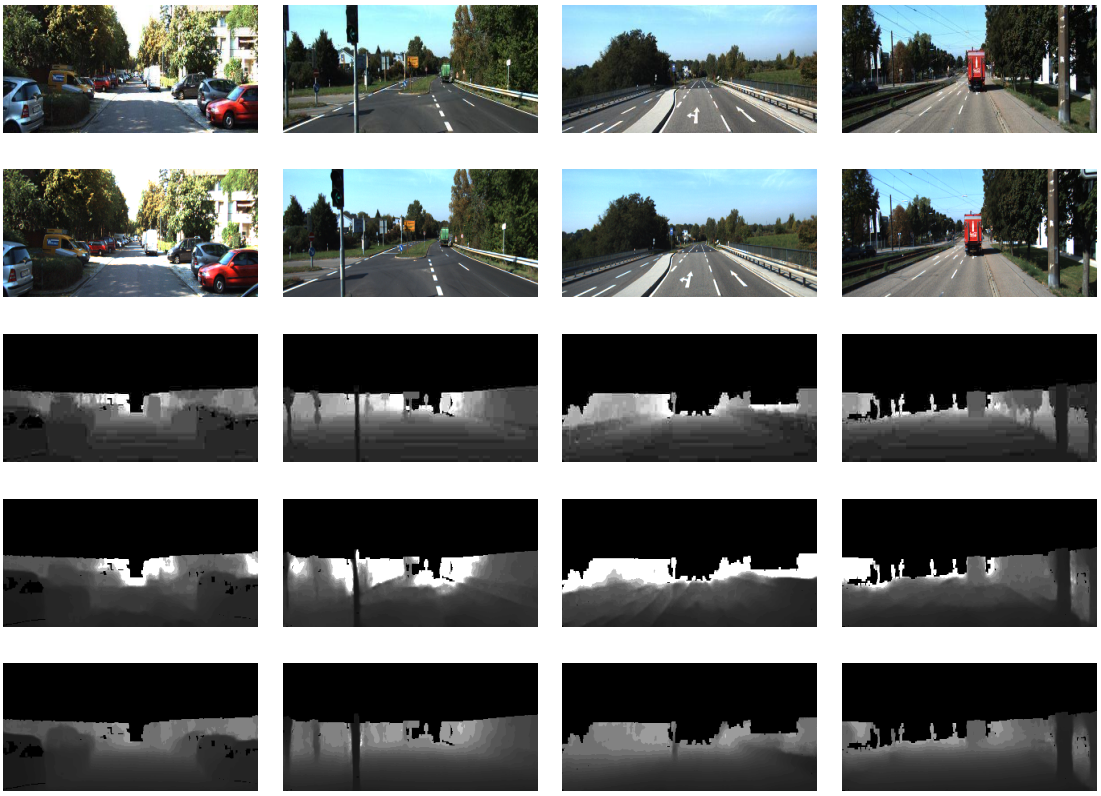
The depth quality of the former is better refined, sharper with less noise and more inherent details especially along the separation lines between objects.
The current observations contradict indeed the assumption that depth estimation gets profit from learned segmentation features. It also contradicts the previous results with the multi-tasking model for disparity planes estimation. In fact, the joint training in this experiment negatively affects the quality of the estimated output. Important details in comparison to the original results are lost. Depth Maps produced by our model are slightly noisier and depict less details which affects the perceived objects shapes.
This performance can be explained by different causes. To our understanding, the main aspect to consider for analyzing this is the fact that only synthetic data has been used during the training process whereas the testing is performed on real images. The signal characteristics and the properties of the images are obviously way different for the model between test and training phases. Pixel inter-correlation in real images is indeed significantly higher for real images, whereas artificial images usually present higher contrast along the edges since they can be considered as discrete samples from a probabilistic generative model. Moreover, the nature of the noise is as well different. This is a further important aspect discussed in [74] and personally experienced when we dealt with real image denoising methods [75]. In reality, this difference in terms of statistical pixel properties between artificially generated images and real-world captured ones is still a wide open research topic. This became more relevant after the breakout of DL techniques requiring abundant amount of data and ground-truth that might not be always available for real-world scenarios.
All these discussed issues seemed to have minor effects on the performance of the original DispNet. However, our multi-tasking model delivered decent results in terms of depth quality but unexpectedly didn't improve upon the original model. This means that the addition of the segmentation track under the previous conditions does not benefit the learning process. The conclusion is then that semantic segmentation unlike depth estimation is less suitable to such passage from synthetic data to real one between training and testing phases. Consequently, the combination of loss terms into a single one results in the convergence towards a worse minimum compared to the original case of the single DispNet.
A numerical comparison presented in Tab. 3.3 and further visual results in Fig. 3.8 absolved on KITTI images with available ground-truth labels support the previous claim and confirm the same observation that DispNet does not outperform the multi-tasking model just in terms of visual quality but also numerically (almost all measures).
In order to avoid using synthetic training data, we decide to focus on using only real images for training CNNs in the future. However, getting reliable depth ground-truth for such data is difficult. That is why we decide to look into other alternative ways to perform depth estimation without opting for training on artificial images. Once this is achieved, the examination of adding semantic segmentation tracks within multi-tasking models can be again fairly evaluated. What alternative methods for CNN-based depth estimation could be considered to realize this?

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Single DispNet model	0.4014	11.4844	23.6715	0.7496	0.2672	0.2592	0.3764	0.5214
Our Multi-tasking model	0.4213	12.1394	24.3305	0.7784	0.2823	0.2323	0.3724	0.5088

Lower is better

Higher is better

Table 3.3: Numerical evaluation on the test Eigen-split set (697 image) of KITTI [11] with resolution (572×160) of both single and multi-tasking models for supervised depth estimation



(a) 1.Scene (b) 2.Scene (c) 3.Scene (d) 4.Scene

Figure 3.9: Visual Evaluation on KITTI Eigen-split Test Set: Comparison of visual results using single and multi-tasking models for supervised depth estimation. From top: Left image, right image, ground-truth depth map for left view, estimated left depth map using single DispNet, estimated left depth map using our multi-tasking model.

4 Multi-tasking With Unsupervised Applications: The Segmentation Effect

4.1 Motivation & Related Works

Although DispNet [72] is the state-of-the-art CNN-based method for supervised stereo depth estimation, its main disadvantage is that it needs to be trained on artificial data. In fact, the real issue is that such supervised learning approaches require large datasets with high quality ground-truth labels for training. Unfortunately, this requirement is difficult to fulfill since acquiring reliable depth ground-truth for real images is a very tedious task.
To circumvent the problem, certain works suggest to opt for an unsupervised training strategy where no ground-truth is required. For instance, Xie *et al.* present the Deep3D network [76] that uses a customized reconstruction loss for each pixel based on the distribution over potential disparity candidates. In fact, the combination of the weighted sum of the disparity candidates' probabilities on the same line boils down to the estimated right image pixel. The main drawback of this approach is that complexity is tightly related to the increasing number of disparity candidates. Moreover, Zhou *et al.* introduced in [77] an unsupervised learning framework for the task of monocular depth and camera motion estimation from unstructured video sequences. The approach synthesizes single-view depth and camera pose estimation using two separate networks that are coupled by a common loss based on warping nearby views to the target. Besides, Ravi Garg *et al.* [78] trained a CNN architecture where the loss function describes the photometric reconstruction error between a rectified stereo pair of images. The disparity is replaced by its first order Taylor expansion in order to simplify the gradient computation during the back-propagation step. Godard *et al.* [6] use a fully derivable spatial transform network module to circumvent the problem of gradient computation during back-propagation. Their final loss function is indeed more complex than the simple alignment loss used in [78]. It has additional terms for smoothing and enforcing left-right consistency to improve the convergence. Moreover, Kuznietsov *et al.* [79] propose a hybrid approach for tackling the issue of real data shortage for monocular depth estimation. This semi-supervised approach combines training using an appearance matching loss similar to the one suggested in [6] and a supervised objective function using sparse ground-truth labels acquired from LIDAR sensors. In [80], the depth ground-truth is automatically derived from simple videos recorded by moving cars. They deploy classic depth from motion techniques to avoid human intervention in the training process of the agent learning raw ego-motion. They claim that the proposed model induces features in the network that results in large improvements over a network trained from scratch in many

downstream tasks such as semantic segmentation, joint road segmentation, car detection and monocular (absolute) depth estimation.

4.2 Unsupervised Depth Estimation: UMDELR

4.2.1 Introduction

In order to circumvent the problem of lacking reliable ground-truth data, an alternative way of training is necessary to replace the supervised learning. Hence, unsupervised CNN-based methods for depth/disparity estimation are gaining now more attention. Obviously, the acquisition of accurate depth/disparity ground-truth is much more difficult in comparison to acquiring a pair of stereo rectified images. The idea shared by most approaches mentioned above is the fact to bypass the supervised loss calculation by replacing it with a new loss term. This new loss function needs to depend only on the a rectified stereo pair given as input. As it is currently the best performing state-of-the-art model in the context of unsupervised depth estimation, we consider the model proposed by Godart *et al.* in [6] for further examination.

4.2.2 Implementation Details

As a start, Let us take a closer look at the main differences between state-of-the-art supervised disparity/depth estimation e.g. DispNet [72] and unsupervised methods such as the one by Godart [6]. We focus on the aspect of loss definition and calculation.
In the case of supervised disparity estimation, the training of the model requires a pair of rectified stereo images I_l and I_r as input with their corresponding ground-truth disparity maps D_l and D_r. We are just interested in the learning of left disparity map D_l.
For the sake of simplicity, we restrict ourselves to the example of basic L1 loss calculation at a pixel location (x,y). For the supervised case, it can be written as:

$$L_1(x,y) = ||D_l(x,y) - \hat{D}_l(x,y)|| \tag{4.1}$$

with $\hat{D}_l(x,y)$ is the ground-truth disparity value and $D_l(x,y)$ is the estimated value by the network at pixel location (x,y). $L_1(x,y)$ is the error value at the same pixel location. The goal of the supervised training is to minimize this error between estimated and ground-truth disparity values over all pixels in the different images available in the training set.
However, the mathematical description of the optimization problem in the case of unsupervised depth-disparity estimation is different. The rest of the used framework is almost unchanged since training the network requires a given stereo pair as input, while output is the estimated disparity map. Obviously, the difference revolves only around the definition of the loss function used during the training. As mentioned, the residual is no longer dependent on a ground-truth data but involves only the input images and the estimated

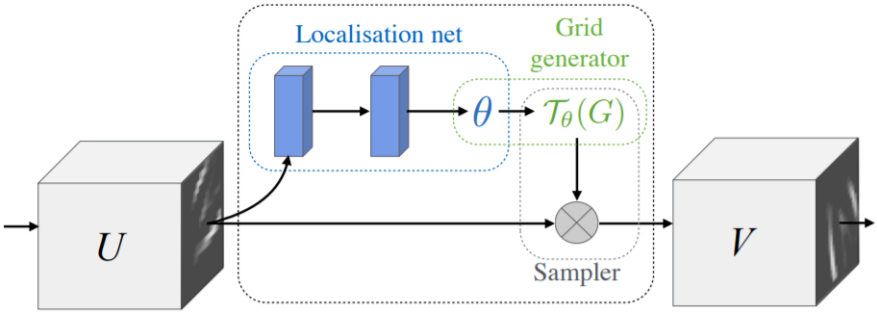


Figure 4.1: The spatial transformer module (courtesy of [5])

output disparity of the network relating both. In other words, if the network gives as output an estimated D_l relative to left image view, this estimated disparity is then used to wrap the right view into the left one. Ultimately, the error function to optimize is calculated between the resulting wrapped image $\hat{I}_l$ and the already available original left image I_l. It can be written as:

$$L_1{}'(x,y) = ||I_l(x,y) - \hat{I}_l(x,y)|| = ||I_l(x,y) - I_r(x + D_l(x,y),y)|| \tag{4.2}$$

Nevertheless, it is obvious that $\hat{I}_l(x,y) = I_r(x + D_l(x,y),y)$ is not derivable in respect with $D_l(x,y)$, which makes it consequently unsuitable to CNN training. In fact, the back-bone step for training using gradient descent approaches is the back-propagation step that requires gradient computation for propagating the error through the network layers up to the early ones.
In order to tackle this obstacle, Godart *et al.* proposed in [6] a model inspired from DispNet. Their model comes additionally with a locally differentiable loss module for unsupervised learning. The module samples the left image from pixels of the opposite stereo image using an image sampler introduced in [5] as a part of the spatial transformer network (STN). Jaderberg *et al.* proposed the STN, a learnable module that allows the spatial manipulation of data within neural networks. It performs explicit spatial transformations of features in an end-to-end way without modifying the loss function. It can be inserted into existing architectures as a complete module involving three main blocks: A localization network, a grid generator and an image sampler. An overview of the STN details from [5] can be seen in Fig. 4.1.

The shown architecture of the spatial transformer module consists of three separable building blocks. First, the localization network receiving the feature map U as an input before the spatial modification. It regresses the applied transformation T_θ and describes it using θ. This variable varies depending on the parametrized transformation type, e.g. for an affine transformation θ should be 6-dimensional as in equation. 2.1. Second, the parametrized sampling block is responsible for transforming the regular spatial grid G over wrapped output feature map V to the sampling grid $T_\theta(G)$ applied on incoming feature map U (see Fig. 4.2). Each target pixel (x_i^t, y_i^t) lying on the regular grid G of feature map

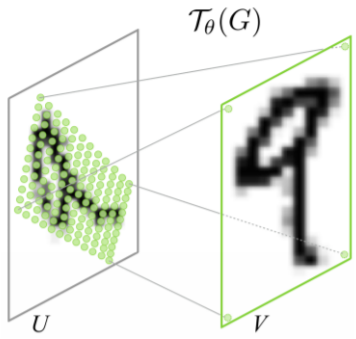


Figure 4.2: The sampling grid is the result of warping the regular grid G with an affine transformation $T_\theta(G)$ (courtesy of [5])

V is computed by applying a sampling kernel centered at a particular location (x_i^s, y_i^s) in the input feature map U. If the applied transformation T_θ is a 2D affine transformation described by matrix $A_\theta \in R^{2\times 3}$, the pixelwise transformation can be written as:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(G) = A_\theta \cdot \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \cdot \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{4.3}$$

The final building block of the STN module is the differentiable image sampler. The sampler takes as input the set of sampling points $T_\theta(G)$ along with the input feature map U and generates as output the sampled map V. Every single pixel location (x_i^s, y_i^s) on the sampling grid $T_\theta(G)$ is considered as the application location of an operation realized by a kernel centered on this same location. The output of this operation defines the resulting value at the particular pixel (x_i^t, y_i^t) in the feature map V. In the case of a bilinear sampling operation, we can write:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \tag{4.4}$$

where U_{nm}^c is the value at location (n,m) in channel c of the input feature map U and V_i^c is the output value for pixel i at location (x_i^t, y_i^t) in channel c of feature map V. H and W are respectively height and width of input map U.
The advantage of this term is that it is locally fully differentiable and integrates seamlessly into our fully convolutional architecture without simplification or approximation of the cost function. (Sub)-gradients with respect to source coordinates (x_i^s, y_i^s) in input grid G and values of feature map U passed to the sampler can be written as:

$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c max(0, 1-|y_i^s - n|) \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

$$\frac{\partial V_i^c}{\partial y_i^s} = \sum_n^H \sum_m^W U_{nm}^c max(0, 1-|x_i^s - n|) \begin{cases} 0 & \text{if } |m - y_i^s| \geq 1 \\ 1 & \text{if } m \geq y_i^s \\ -1 & \text{if } m < y_i^s \end{cases}$$

Unlike DispNet, the main specificity of the UMDLR model is the fact that it poses the depth estimation problem as a ground-truth-free reconstruction problem. It does not only consider the minimization of a basic photometric loss, as discussed previously (see equation. 4.2), but proposes additional left-right consistency terms and smoothing constraints in order to refine the quality of the estimated output depth.
The architecture of the network has an encoder-decoder layout. Two architecture versions are indeed proposed. The first one uses a VggNet-based [81] encoder and the second uses a ResNet-based encoder [34]. Both versions use the same customized depth estimation loss based on STN. During the training phase, ground-truth depth is no longer required. Only a pair of calibrated binocular images is needed to perform the loss computation. The encoding parts of both versions have successive spatial convolutions. The decoders are always symmetrical with the same number of transposed convolutional layers. Both parts are connected through long range links. At four different scales, disparity outputs (left-to-right D_l and right-to-left D_r) of the given input stereo images are estimated simultaneously. The dimensions double in a subsequent order from a scale level to the next.
Using the image sampler of the previously discussed spatial transformer module, the network learns at each scale to generate the estimated wrapped images $\hat{I}_l$ and $\hat{I}_r$ by sampling pixels from the respective opposite images I_l and I_r based on the estimated disparities. The resulting fully differentiable loss function of the model can be summarized as:

$$C = \sum_{s=1}^{4} C_s = \sum_{s=1}^{4} \alpha_{ap}(C_{ap}^l + C_{ap}^r) + \alpha_{ds}(C_{ds}^l + C_{ds}^r) + \alpha_{lr}(C_{lr}^l + C_{lr}^r) \tag{4.5}$$

with $\alpha_{ap} = \alpha_{ds} = \alpha_{lr} = 1$ are weighting factors to the different terms and $s = 1..4$ is the corresponding scale. The different terms combine left and right contributions to the calculation of the final loss that can be introduced as follows. For the sake of simplicity we restrict the description to the left image. For the right image, it is calculated similarly except for corresponding index changes:

- Appearance matching Loss:

$$C_{ap}^l = \frac{1}{N} \sum_{i,j} \delta \frac{1 - SSIM(I_{ij}^l, \hat{I_{ij}^l})}{2} + (1-\delta)||I_{ij}^l - \hat{I_{ij}^l}||$$

for photometric similarity with N number of image pixels. $\delta = 0.85$ a weighting factor and SSIM is a single scale distance measure introduced in [82] with a 3×3 block filter instead of the Gaussian filter originally used.

- Disparity Smoothness Loss:

$$C_{ds}^{l} = \frac{1}{N} \sum_{i,j} |\partial_x D_{ij}^{l}| \cdot e^{-||\partial_x I_{ij}^{l}||} + |\partial_y D_{ij}^{l}| \cdot e^{-||\partial_y I_{ij}^{l}||}$$

for smoothing and refinement by weighting depth discontinuities ∂D with an edge aware term using image gradient ∂I

- Disparity Consistency Loss

$$C_{lr}^{l} = \frac{1}{N} \sum_{i,j} ||D_{ij}^{l} - D_{ij+D_{ij}^{l}}^{r}||$$

for left-right consistency penalty and enforcing both disparities to be similar to their projected versions.

For training the network, the Tensorflow framework [83] a flexible DL library released by Google is used. It offers much more flexibility for gradient computation thanks to its auto-gradient computation feature in comparison to many other frameworks e.g. Caffe. This advantage can alleviate the enormous load of manually programming the gradient calculation of the proposed new loss function necessary to the back-propagation during the training. Considering the reported results by the authors in [6], we opt for training the Resnet50 version of the model with post-processing step. Obviously, this version delivers the best visual and numerical performance on the different test sets considered for evaluation in paper [6] and by us for all discussed experiments that will follow:

- 1. set: The 200 images of KITTI 2015.
- 2. set: The "testing Eigen-split" of KITTI with 697 images.

As for the training sets, we consider for our experiments three different setups. First, the "Trainvaltest" set of Cityscapes consisting of the 3475 merged train and validation images. Second, the "Trainextra" set of Cityscapes consisting of 19998 images combined with the previous set which results in 23473 images in total. Finally, the third set consists of only the KITTI Eigen-split for training with 22600 selected images. To summarize the three experimental training setups we will be using all along this chapter, we can write:

- 1. set: The "Trainavaltest" set of Cityscapes with 3475 images.
- 2. set: The "Trainavaltest" set of Cityscapes with 3475 images + the "Trainextra" set of Cityscapes with 19998 images equals in total 23473 images.

- 3. set: The "training Eigen-split" of KITTI with 22600 images.

For both first cases, all models discussed in the current chapter are trained from scratch using random weight initialization (Gaussian). For the third scenario, we always consider the weights of the final model from case 2 as initial values. In other words, we fine-tune the second case's model on the KITTI training set. In fact, we follow here the recommendation of the authors that assert that fine-tuning on KITTI images gives better convergence than normal training with mixed KITTI and Cityscapes images from the beginning.
As for data augmentation, horizontal flipping of the input images was performed on the fly following a 50% probability. Random gamma, brightness, and color changes sampled from uniform distributions from the ranges: $[0.8, 1.2]$ for gamma, $[0.5, 2.0]$ for brightness, and $[0.8, 1.2]$ for each color channel separately are as well considered. As optimizer, Adam [52] was adopted. The model is trained for 50 epochs with batch size equals 8, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ and $\lambda = 10^{-4} =$ divided by 2 after every 10 epochs starting from 30 epochs.

4.2.3 Results

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1.Experiment	0.299	4.095	9.648	0.329	70.987	0.577	0.848	0.951
2.Experiment	0.206	2.962	8.632	0.303	48.918	0.736	0.881	0.944
3.Experiment	**0.0853**	**1.034**	**4.890**	**0.160**	**16.274**	**0.918**	**0.972**	**0.987**

(a) Numerical Evaluation on the KITTI 2015 test set (200images)

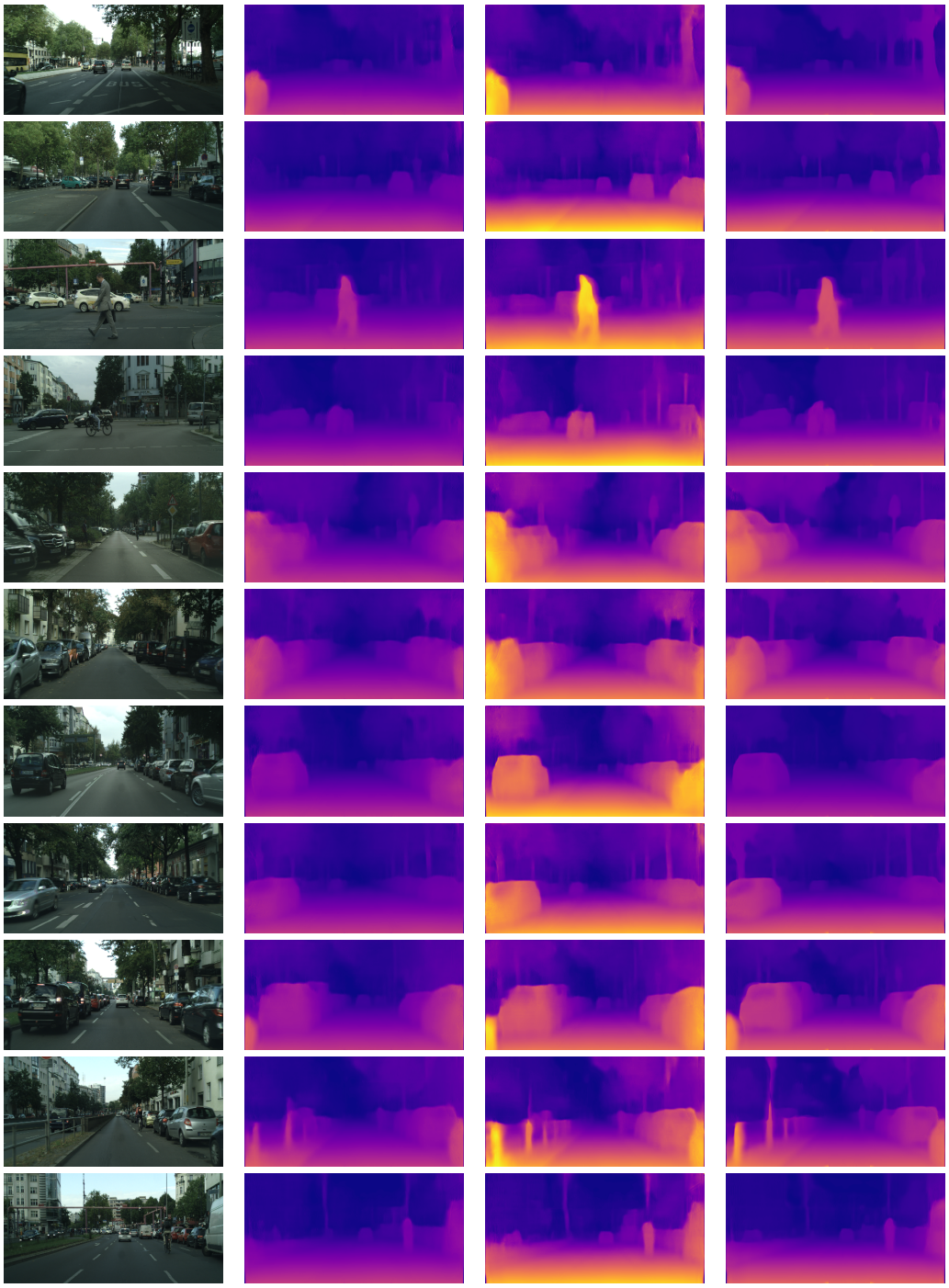
Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
1.Experiment	0.275	3.463	8.916	0.331	0.610	0.845	0.941
2.Experiment	0.236	2.848	8.467	0.358	0.660	0.840	0.920
3.Experiment	**0.1217**	**1.011**	**5.221**	**0.218**	**0.852**	**0.943**	**0.973**

(b) Numerical Evaluation on the Eigen-split test set (697images)

Lower is better
Higher is better

Table 4.1: Numerical performance comparison of the UMDLR original architecture trained in three different experimental setups with different training sets

Three different experiments has been conducted including the training of the same network architecture with a different set each time as detailed above. The weight initialization is random for the first and second experiments while the third experiment is fine-tuned version of the second model on the KITTI set. The numerical performance evaluation of the three models is detailed in Tab. 4.1.
Obviously, the extension of the training set improves significantly the performance by each experiment. The more data is given the better is the learning outcome. However, this improvement is less noticeable in the third experiment since the used data for fine-tuning



(a) Input frame (b) 1.Exp Model's result (c) 2.Exp Model's result (d) 3.Exp Model's result

Figure 4.3: The visual performance of the UMDLR model trained on different training sets ((a) 1.Experiment (b) 2.Experiment (c) 3.Experiment) on test images of Cityscapes.

present different semantic content and other characteristics than the used test set (Fine-tuned on KITTI and tested on Cityscapes). Nevertheless, the improvement of the estimated depth masks in terms of refined details and inherent noise of the third model proves that enriching the training set with rich reliable data can only boost the performance for DL applications including the unsupervised depth estimation. This is clearly shown in Fig. 4.3 where output results of the three models are displayed together with input images from the Cityscapes test set. The gradual improvement by each experiment is noticeable in terms of estimated depth quality.

4.3 Cooperative Optimization For Unsupervised Depth Estimation And Semantic Segmentation

4.3.1 Introduction

After discussing the model used for unsupervised depth estimation, we propose a multi-tasking model to jointly learn unsupervised depth estimation and semantic segmentation within a cooperative multi-objective optimization framework as we did with experiments discussed in the previous chapter with supervised applications (supervised depth and disparity planes estimation).

4.3.2 Proposed Approach

As with the supervised case, the multi-tasking model we propose for this experiment is based on the scalarization of multiple objective functions. Its architecture has two tracks sharing a common encoder of Resnet50 style. The decoders are however different. Each one is responsible for recovering the details for a specific task. The first one is of Resnet50 style and it ends with the unsupervised depth loss detailed in the previous section. The second decoder is borrowed from the MokaNet architecture used for semantic segmentation. An overview of the whole architecture is shown in Fig. 4.4.
The proposed model requires a stereo pair as input. The output is also a pair consisting of the semantic segmentation mask and the disparity map of the left input view. Training the network requires the simultaneous availability of the right view necessary to the estimation of pixel-wise displacements between images and a semantic segmentation ground-truth label used for learning proper classification.
Unfortunately, the considered training sets, detailed previously, do not always offer right image views and segmentation ground-truth labels at the same time except for the first case (1.Experiment). Therefore for our current model, we restrict our experimentation on the first training setup. We just consider the merged 3475 images of the Cityscapes "Trainvaltest" set in order to train the multi-tasking cooperative model. Consequently, a fair comparison between the performance of single UMDLR and our multi-tasking model (the

UMDLR combined with segmentation) could be guaranteed.
For training, we replicate the original environment used by the authors in [6] together with the exact same data augmentation applied online. We apply horizontal flipping to the input images with 50% probability. Photometric transformations including random gamma, brightness, and color variations by sampling from uniform distributions within preset ranges: $[0.8, 1.2]$ for gamma, $[0.5, 2.0]$ for brightness, and $[0.8, 1.2]$ for each color channel separately are employed. Additionally, Adam optimizer [52] was again used for training the model for 50 epochs from scratch with a batch size of 8, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The starting learning rate $\lambda = 10^{-4}$ is kept constant for the first 30 epochs before halving it every 10 remaining epochs.

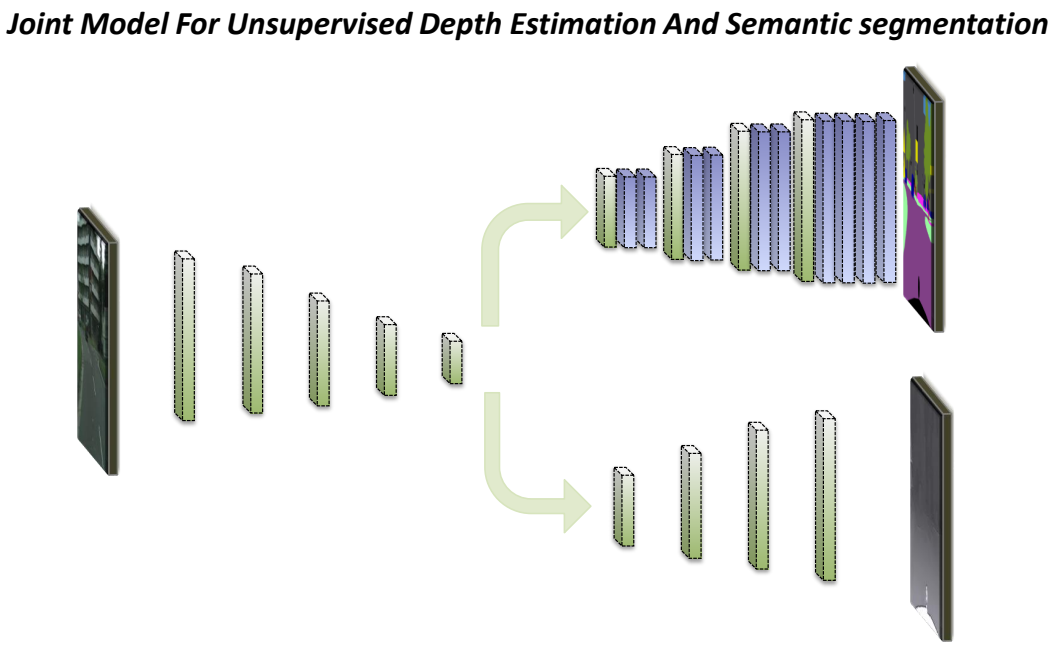
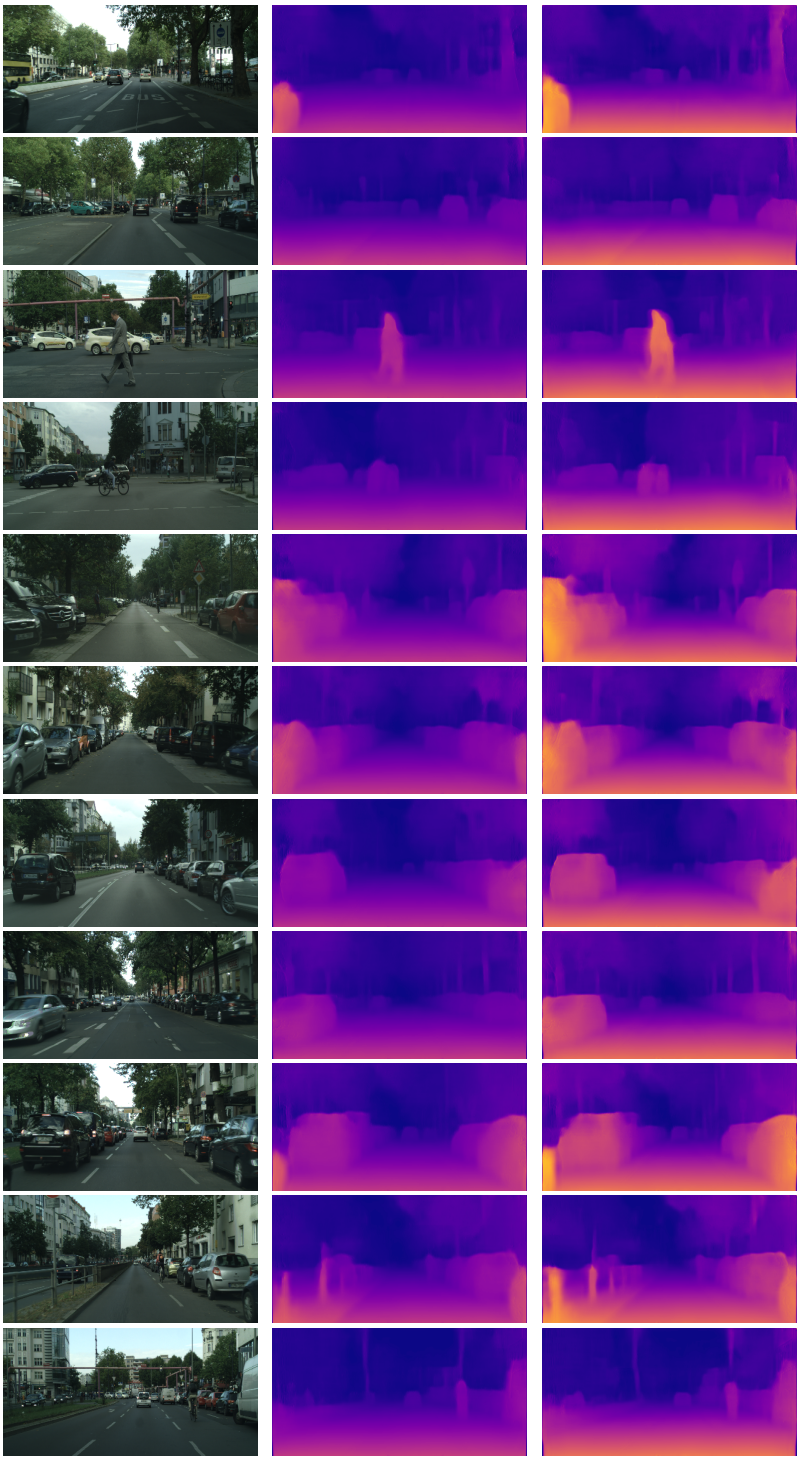


Figure 4.4: The multi-tasking model architecture for joint unsupervised depth estimation and semantic segmentation

4.3.3 Results & Discussion

In Fig. 4.5 and Tab. 4.3, we summarize the numerical and visual aspects of the performance comparison between the original UMDLR and the multi-tasking UMDLR model that additionally profited from the simultaneous learning of semantic segmentation features. Considering the exact same training conditions (including data augmentation, training set



(a) Input frame

(b) Estimated depth map of the UMDLR model (1. Experimental setup)

(c) Estimated depth map of the cooperative multi-tasking model (1. Experimental setup)

Figure 4.5: The visual comparison between the cooperative multi-tasking and the original UMDLR models deployed on test images of Cityscapes.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.299	4.095	9.648	0.329	70.987	0.577	0.848	0.951
Cooperative	**0.239**	**2.950**	**8.647**	**0.313**	**60.918**	**0.747**	**0.873**	**0.942**

(a) Numerical Evaluation on the KITTI 2015 test set (200images)

Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.275	3.463	8.916	0.331	0.610	0.845	0.941
Cooperative	**0.231**	**3.219**	**8.441**	**0.307**	**0.688**	**0.884**	**0.957**

(b) Numerical Evaluation on the Eigen-split test set (697images)

Lower is better
Higher is better

Table 4.3: The numerical comparison between the cooperative multi-tasking model and the original UMDLR model. (Experimental setup 1)

and optimizer parameters, etc...), the multi-tasking UMDLR model outperforms the original one in terms of produced depth quality. Visually, results on the test images of Cityscapes show a significant improvement between both models. The effect can be noticed on the refined object shapes and the clear contours as well as in the additional far details. Moreover, a clear attenuation of the undesired blurriness (extreme smoothness) due to the used bilinear filtering from the STN module is noticed. The estimated depth output seems indeed sharper with better visible details and more accurate approximated distance information.
Numerically, the same observation can be drawn from Tab. 4.3a and Tab. 4.3b. The joint UMDLR model outperforms the single one in all considered metrics when evaluated on the 200 images of KITTI 2015 as well as on the 697 images of the testing KITTI Eigen-split. Ultimately, this experiment proves again the assumption previously made concerning the positive effect of *transfer learning* between different simultaneously trained applications within a multi-tasking cooperative CNN. This mutual inductive exchange of features learned simultaneously helps reduce the scope of possible bias in a beneficial way by forcing the model to fit different but related tasks. In our case, unsupervised depth estimation and semantic segmentation seem to be enough related applications. In reality, this limit defining, how different versus how related two tasks should be, is a very unexplored terrain. It is still a wide discussion topic within the community. From our experiment however, we can deduce that the relation between unsupervised depth estimation and semantic segmentation lies within the range of the spectrum necessary to guarantee the profitability between these two tasks. More precisely, we focus here on the positive effect of high quality segmentation on unsupervised estimated depth.
A weak aspect of the current experience can not be neglected. This is the necessity to have right views and segmentation ground-truth labels available at the same time. This case seems less challenging than requiring segmentation and depth ground-truth simultaneously. The problem however here is the size of the available data pool. In general, requiring a single homogenous dataset offering the possibility to learn all wanted tasks (available ground-truth labels for all tasks) at once is always a restriction. Either there will

be no data available at all or the size won't be enough for training robust CNNs.
Despite the abundant amount of data available out there for both our current applications, it was not possible at the time of implementation to find a dataset that offers us the needed setup. As an example, we mention the illustrative case of the most popular KITTI and Cityscapes datasets. The former is generally used for depth estimation purposes and the latter is considered for learning semantic segmentation. Learning both at the same time using a single dataset is not possible. So how can we get simultaneous profit of such heterogeneous datasets using the same model?

4.4 Competitive Optimization For Unsupervised Depth Estimation And Semantic Segmentation

4.4.1 Motivation & Background

Facing the necessity of providing right view images and semantic segmentation ground-truth labels simultaneously during training, we were restricted to train our multi-tasking co-operative UMDLR model only using the training set of the first experimental setup. Among the three detailed scenarios, such experiment is only possible using the "Trainvaltest" of Cityscapes.
This restricting issue drove us to consider the optimization problem at hand from an other perspective. If we don't have enough simultaneous ground-truth available for both tasks, why not modify the training procedure and make it more flexible by accepting inputs from different datasets not necessarily having the same label? In other words, why not focusing on flexible ways to train the network as introduced in [84] instead of focusing on requiring proper data at all costs?
In fact after opting for a scalarizing technique to jointly solve the multi-objective optimization problem of simultaneous unsupervised depth estimation and semantic segmentation, we decide to address the problem from a different point of view by performing optimization of the same set of parameters using different input datasets [84]. In reality, few other arguments presented in [85] concerning the objective scalarizing method added up to the motivation behind changing our strategy: First, the non-trivial choice of weighting factors $\alpha_i, \forall i = 1, .., n$ in equation. 3.7, that have an important effect on the optimization behavior and the resulting potential optimum towards which the problem converges. Second, the optimization of a saclarized objective function delivers a single solution for all tasks. If these tasks are conflicting, which is very likely if the nature of the applications are different, this single solution optimizing all the tasks can not be simultaneously optimal for all of them. Taking into consideration these points, we look for an alternative approach to perform multi-objective optimization with CNN models.
As a starting point, we borrow few concepts known from the game theory and the strategy field, namely the concept of Nash non-cooperative games [86] in order to better analyze the optimization problem at hand. For this, let us consider the multi-objective optimization

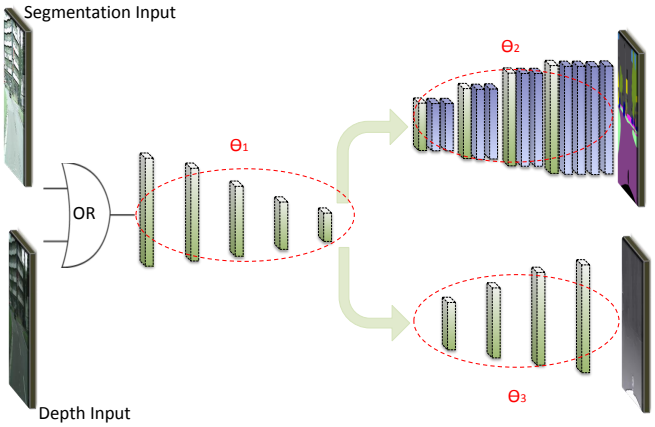
task described by equation. 3.5 as a game played by two players having each a corresponding cost function to optimize independently. Both players should have access to the same parameter space and their motivation is to minimize separately their cost functions (elements of the vectorized loss function) while partially taking into consideration the other player's decision.
Instead of summing up the separate objectives in a single loss term, the first element of the vectorized loss function in our experiment is assigned to the unsupervised depth estimation agent/track (player 1), and the second element to the semantic segmentation agent/track (player 2). The game (the optimization procedure) is designed as a competition between players that do not cooperate to minimize a common objective but alternate to minimize separate ones. For each iteration, a single player is allowed to play (update the weights of the shared parameter space) in order to win (minimize its corresponding loss term). In other words, the players are no longer cooperating together to win the game, as it is the case with the scalarizing method, but they are competing against each other while still willing to win the same game.
To summarize, our designed game is an optimization approach simulating the interaction of two agents (players). Both have the goal to optimize their own gain by following their own strategies taken into consideration partial knowledge about the strategy of the opposing player. This is called a competitive (non-cooperative) game. It starts with the first player making a decision. Subsequently, the second player takes his own decision based on the status of the game after his opponent's previous move. Then the game goes on as a recurrent repetition of this alternating pattern.

4.4.2 The Proposed Approach

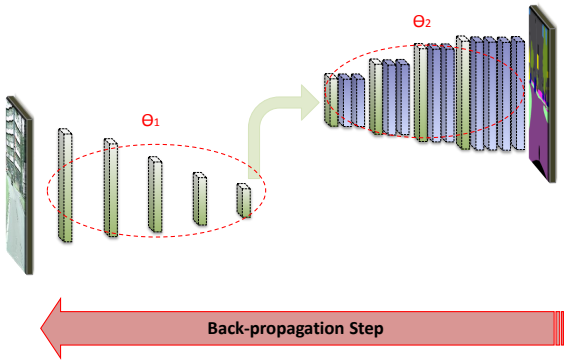
For our optimization problem, the two objective functions $L_1(\theta)$ and $L_2(\theta)$ correspond to semantic segmentation and unsupervised depth estimation respectively. The goal is to find the optimal set of parameters θ that minimizes both functions. Unlike our previous strategy of combining both losses into a single objective function, the new approach we propose requires that each player updates a part of the parameters' vector θ. Each one separately follows his own minimization strategy while taking into consideration the other player's decision. As shown in Fig. 4.6, if we formulate the problem mathematically, we can write the learnable parameters' vector θ as a combination of three sub-vectors:

$$\theta = (\theta_1, \theta_2, \theta_3)$$

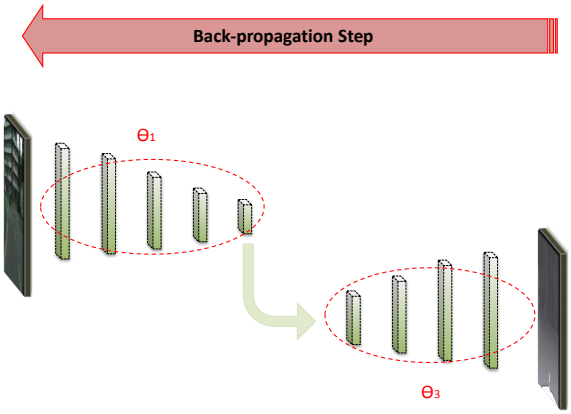
θ_1 is the sub-vector containing the weights of the layers in the shared encoder part, θ_2 is the sub-vector with weights for layers forming the decoder part responsible for semantic segmentation and θ_3 for the decoder part responsible for unsupervised depth estimation. Since two separate tracks are considered, agents cannot update the weights of the part responsible for the other task. This means that only the relevant decoder and the shared encoder weights can be updated by a single player on each iteration (each player cannot



(a) Overall Architecture for the used model



(b) Activated track1 for semantic segmentation (player1)



(c) Activated track for unsupervised depth estimation (player2)

Figure 4.6: Conditional training procedure for CNN-model to solve non-cooperative optimization of unsupervised depth estimation and semantic segmentation problems

see the weights of the decoder in the track of his opponent). Vectors $A = (\theta_1, \theta_2)$ and $B = (\theta_1, \theta_3)$ could be considered as the strategies in a symmetrical Nash game with the final goal to optimize $L_1(\theta)$ and $L_2(\theta)$ respectively. Each player uses a subset of θ to minimize its cost function: A for player 1 and B for player 2.
The point at which the game is expected to end (the optimization is done) is called the Nash equilibrium [86]. this state is characterized by the vector $\bar{\theta} = (\bar{\theta}_1, \bar{\theta}_2, \bar{\theta}_3)$ responsible for this equilibrium between both criteria and defined as:
Nash Equilibrium: $\bar{\theta} = (\bar{\theta}_1, \bar{\theta}_2, \bar{\theta}_3)$ is called to realize a Nash equilibrium between criteria $L_1(\theta)$ and $L_2(\theta)$ if and only if (iff)

$$\begin{cases} (\bar{\theta}_1, \bar{\theta}_2) = \underset{\theta_1, \theta_2}{\arg\min} & L_1(\theta_1, \theta_2, \bar{\theta}_3) \\ (\bar{\theta}_1, \bar{\theta}_3) = \underset{\theta_1, \theta_3}{\arg\min} & L_2(\theta_1, \bar{\theta}_2, \theta_3) \end{cases} \tag{4.6}$$

According [86], the equilibrium solution is only influenced by the split of the parameters vector. This is commonly referred to by using the concept of "split of territory". As detailed in [87], this important aspect of the game describes the way the parameter space is divided between players. In other words, which region of the parameter space is allocated to which player in order to apply its moves (please refer to [86], [87], [88] for more details).
In our game, the design of the architecture defines this aspect. Obviously, it affects the solution at which the Nash equilibrium is reached. It is the most crucial aspect of the reformed optimization problem. Moreover, the training procedure needs to be revised for the new problem. It can be summarized as follows:

1. Design network with learnable parameters' vector θ and initialize its sub-vectors as $\theta^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)})$

2. Activate semantic segmentation track: Freeze $\theta_3 = \theta_3^{(0)}$ and optimize $L_1(\theta_1, \theta_2, \theta_3^{(0)})$ by applying a back-propagation step for the semantic segmentation track. Just weights of the sub-vector (θ_1, θ_2) are updated. The learnable parameters vector becomes $\theta^{(1)} = (\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(0)}))$.

3. Activate unsupervised depth estimation track: Keep $\theta_2 = \theta_2^{(1)}$ and optimize $L_2(\theta_1, \theta_2^{(1)}, \theta_3)$, by applying a back-propagation step for the unsupervised depth estimation track. Just weights of the sub-vector (θ_1, θ_3) are updated. The learnable parameters vector becomes $\theta^{(2)} = (\theta_1^{(2)}, \theta_2^{(1)}, \theta_3^{(2)}))$.

4. Keep alternating between step 1. and 2. and updating sub-vectors correspondingly until Nash equilibrium is reached. If track 1 is making the best possible decision (track 1 converged) taking into account track 2's decision while track 2's decision remains unchanged (track 2 converged), and track 2 is making the best possible decision (track 2 converged) taking into account track 1's decision while track 1's decision remains unchanged (track 1 converged)), then the competitive optimization is done.

For the training, a similar training setup to what has been used during the previous experiment is adopted (framework, data augmentation, optimizer parameters, etc...). Unlike the case of cooperative optimization, the new proposed conditional training strategy allows us to approach all three experimental setups previously detailed. The data for both tracks is not needed simultaneously anymore. To train the second track (unsupervised depth), all three training sets discussed above can be considered. For training the first track, we use every time the available segmentation training set of Cityscapes with 3475 image with corresponding ground-truth labels. For the three experiments, we use the same data augmentation discussed previously with the joint UMDLR model for track 2. For track 1, the semantic segmentation training set undergoes the exact same augmentation operations we suggested in chapter 2 (see section 2.3.1).
To summarize the current experimental setup without considering data augmentation performed on the fly, we can write:

- Experiment 1: Player 1: The "Trainavaltest" set of Cityscapes with 3475 images (semantic segmentation).
  Player 2: The "Trainavaltest" set of Cityscapes with 3475 images (unsupervised depth using right view).

- Experiment 2: Player 1: The "Trainavaltest" set of Cityscapes with 3475 images (semantic segmentation).
  Player 2: The "Trainavaltest" set of Cityscapes with 3475 images + the "Trainextra" set of Cityscapes with 19998 images (unsupervised depth using right view).

- Experiment 3: Player 1: The "Trainavaltest" set of Cityscapes with 3475 images (semantic segmentation).
  Player 2: The training Eigen-split set of KITTI with 22600 images (unsupervised depth using right view).

4.4.3 Results & Discussion

In order to evaluate the effect of the new proposed training strategy, we perform a comparison of the estimated depth outputs of the original UMDLR models (trained under the three different training setups previously detailed) against the new models trained in a competitive way using the new strategy (trained in alternation with a semantic segmentation track). For the first experiment we distinguish three cases. First, we were able to perform a joint training using scalarization as the "Trainvaltest" set of Cityscapes comes with available segmentation ground-truth labels and right view frames necessary for unsupervised depth estimation. Moreover, we managed to train the original UMDLR without obstacles. Finally, we trained it using the new conditional strategy. Consequently, the resulting three models can be compared for the setup of experiment 1:

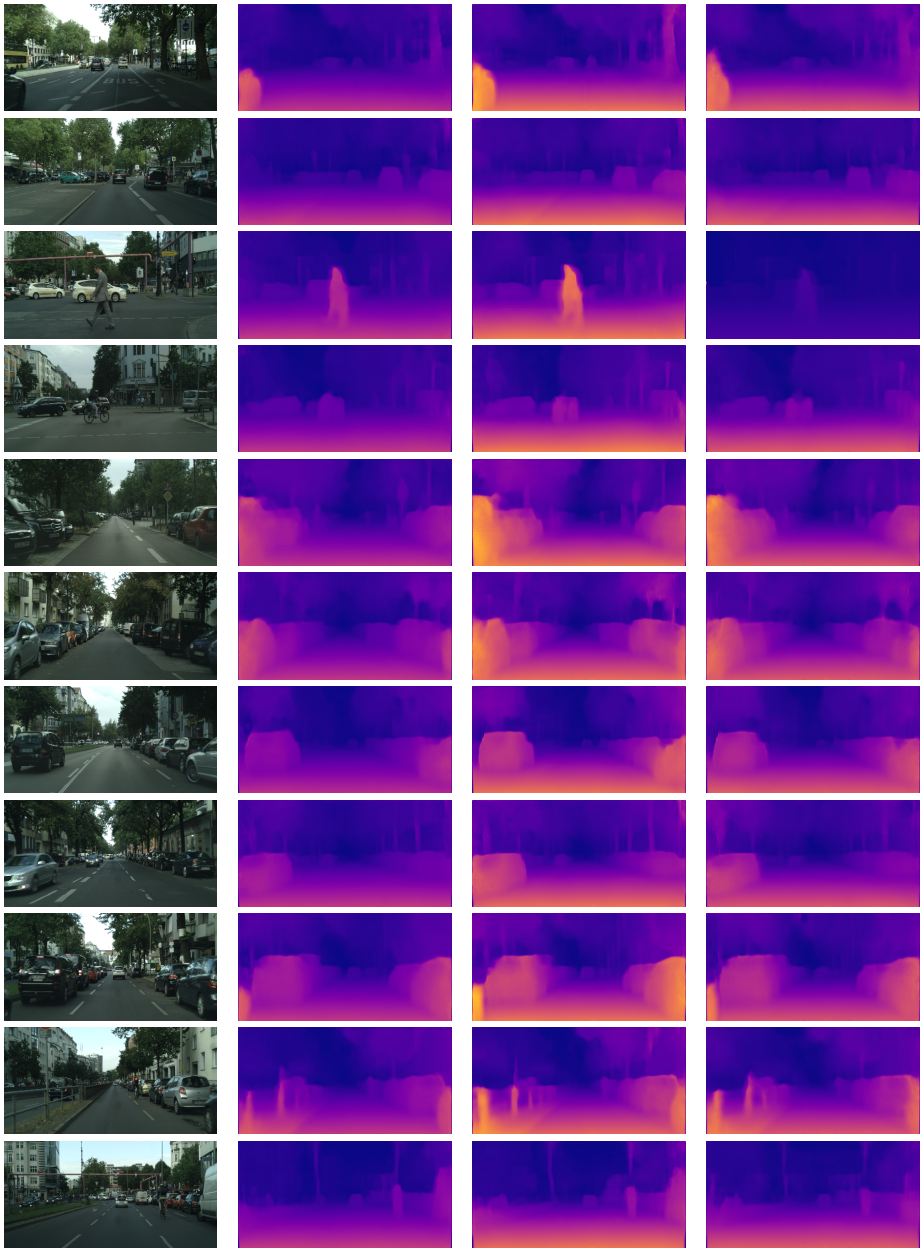
* The original UMDLR model trained independently on the given training set ("Trainvaltest" set of Cityscapes for just depth).

* The cooperative UMDLR model trained jointly/simultaneously with a segmentation track ("Trainvaltest" set of Cityscapes for depth and segmentation).

* The competitive UMDLR model trained in alternation with the semantic segmentation track ("Trainvaltest" set of Cityscapes for depth and segmentation).

A visual survey showing the results of all models is summarized in Fig. 4.7. Output depth maps of images randomly selected from the Cityscapes test set are displayed in parallel for all three scenarios. Obviously, competitive and cooperative models produce better visual depth quality compared to the original model confirming that semantic segmentation features learned simultaneously as well as in alternation can help improve the depth estimation. However, it is very hard to distinguish a clear winner between these two models from visual results.
Therefore, we conduct a numerical comparison between the three models on two different test sets (KITTI 2015 and the testing Eigen-split of KITTI). We present the numbers in Tab. 4.5 that show a significant improvement of all metric values when semantic segmentation is considered during the training. In fact, competitive and cooperative models produce very proximate numbers. Both considerably outperform the original model with a slight advantage to the competitive model especially for the first test set (KITTI 2015). For the second set (testing Eigen-split set), this observation does not totally hold. In fact, the cooperative model overtakes the competitive one in almost all metrics except for squared relative difference and RMSE. This advantage is to be taken with some reservation as the Eigen-split set considered, unlike the first one (KITTI 2015), is generated using a complex process with many interpolation sub-steps due to the sparse quality of the raw given LIDAR ground-truth information. In other words, the ground-truth quality of the first set seems slightly more reliable.
To sum up, we can conclude that both training methods where segmentation is additionally considered result in better quality and more accurate unsupervised depth estimation in respect with the original single UMDLR. Although it is hard to distinguish a clear winner among both, the competitive training strategy appears to be a better alternative for improving unsupervised depth estimation. For a comparable performance, it offers more training flexibility. This could save the user a lot of effort compared to the classical cooperative method, especially if the necessary data is not available for both tasks at once. This is indeed the case of the coming second and third experiments.

For the second and third experiments, the datasets used for depth and segmentation are different which means that using the cooperative strategy to train the model is no longer possible. That is why, only a comparison between the original UMDLR models and the competitively trained ones is realized for both experimental setups. Results of experiment 2 showing a significant improvement of the latter upon the former are displayed in



(a) Input frame (b) 1.Exp mono Model's result (c) 1.Exp cooperative Model's result (d) 1.Exp competitive Model's result

Figure 4.7: The visual comparison between the competitive, the cooperative and the original model depth estimation on test set images of Cityscapes.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.299	4.095	9.648	0.329	70.987	0.577	0.848	0.951
Cooperative	0.239	**2.950**	8.647	0.313	60.918	0.747	0.873	0.942
Competitive	**0.238**	3.208	**8.584**	**0.298**	**59.898**	**0.771**	**0.888**	**0.957**

(a) Numerical Evaluation on the KITTI 2015 test set (200images)

Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.275	3.463	8.916	0.331	0.610	0.845	0.941
Cooperative	**0.231**	3.219	8.441	**0.307**	**0.688**	**0.884**	**0.957**
Competitive	0.239	**3.032**	**8.373**	0.335	0.670	0.863	0.941

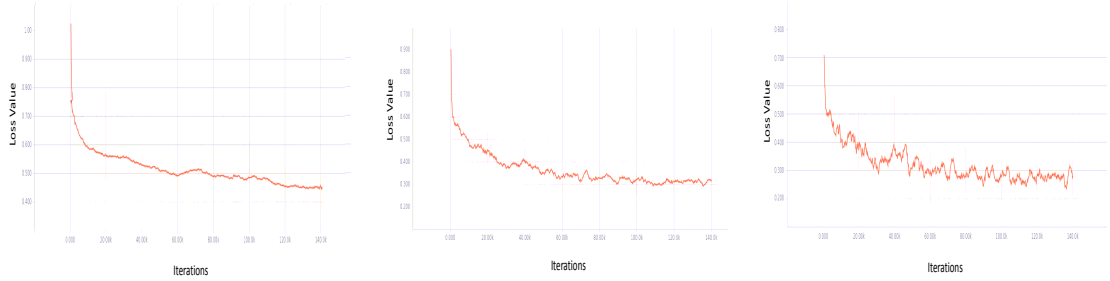
(b) Numerical Evaluation on the Eigen-split test set (697images)

Lower is better

Higher is better

Table 4.5: Experiment 1: Numerical Evaluation

Fig. 4.9 with a clearly enhanced depth quality. Values quantifying the performance on both numerical test sets summarized in Tab. 4.7 show a consistent significant improvement in all metric values when we use the conditional training approach. This improvement is however less significant when compared to the one noticed in experiment 1.
Surprisingly, visual comparison shown in Fig. 4.10 between both models using the third setup (3. Experiment) contradicts this observation as no visible enhancement in the depth quality can be noticed with bare eye. From numbers in Tab. 4.9, we conclude indeed that the original UMDLR model slightly outperforms the competitive one.



(a) Depth loss evolution of competitive model: 1.Experiment

(b) Depth loss evolution of competitive model: 2.Experiment

(c) Depth loss evolution of competitive model: 3.Experiment

Figure 4.8: The comparison of depth loss evolution for the competitive model with the three different experimental setups

In order better analyze these observations, we decide to take a look at the evolu-

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.206	2.962	8.632	0.303	48.918	0.736	0.881	0.944
Competitive	**0.193**	**2.745**	**8.139**	**0.278**	**46.653**	**0.761**	**0.899**	**0.956**

(a) Numerical Evaluation on the KITTI 2015 test set (200images)

Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	0.236	2.848	8.467	**0.358**	0.660	0.840	0.920
Competitive	**0.223**	**2.365**	**7.845**	0.360	**0.669**	**0.845**	**0.923**

(b) Numerical Evaluation on the Eigen-split test set (697images)

Lower is better
Higher is better

Table 4.7: Experiment 2: Numerical Evaluation

tion of the depth loss function of our competitive models in all three experiments. The corresponding results are shown in Fig. 4.8 where the same training interval is always considered for the sake of consistency. We note that all models have a loss curve with descending nature. However, they present differences in terms of smoothness and depicted fluctuations along the complete iterations range. This behavior is noticeable in the second experiment but much more obvious for the third one. A potential cause of this disturbance is related to the significant difference in terms of used training sets. In fact, three different setups are used in each experiment for training the depth track whereas the segmentation keeps the same. To our understanding, the difference between used training sets is not only semantic but also in terms of image characteristics and signal nature. However, the more important aspect is that the difference between datasets used by the two tracks is intensifying from an experiment to the next. In the first one, images for segmentation and depth learning were picked up from the same pool of data ("Trainvaltest" set). This induces better combined feature learning thanks to the enforced redundancy. The inductive transfer between both applications is boosted and significantly enhanced during the training process of experiment one. That is why, we don't see too much disturbance in the depth loss evolution as the passage from an activated track to an other is generally smooth. Ultimately, this smoothly alternating pattern contributes to the improvement of the training procedure in general and the depth estimation in particular compared to the original case. For the second experiment, data from two different sets was picked by each track in every iteration. However, the used training set by the depth ("Trainvaltest" set + "Trainextra" set) and segmentation ("Trainvaltest" set) tracks show strong context similarities and shared properties. Despite consequent content differences, all data comes from the same source (Cityscapes). It was recorded by the same team with the same equipment in very similar conditions and within resembling environments. Therefore, we still notice the effect of the inductive transfer between both applications. This is confirmed by the improved performance in comparison to the original UMDLR trained on the same data. However, the resulting improvement is obviously less considerable than the one noted for the first experiment as the alternation between tracks is less smooth in this case. The slight

fluctuations characterizing the loss evolution compared to the first experiment prove this effect.
In the third experiment, images from Cityscapes were used for learning segmentation whereas the depth track learned from KITTI data. Although both datasets depict traffic scenes from German cities, they still present quite different characteristics. This meaningful difference affected the inductive transfer between both tasks which appears clearly in the sharper fluctuations of the depth loss evolution compared to both previous experiments. As a result, the depth estimation didn't profit from additional segmentation information as much as in previous cases and no considerable improvement upon the original UMDLR model could be noticed when considering this experimental setup.

Method	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	**0.0853**	1.034	**4.890**	**0.160**	**16.274**	**0.918**	**0.972**	**0.987**
Competitive	0.0894	**1.008**	5.097	0.166	17.535	0.912	0.967	0.985

(a) Numerical Evaluation on the KITTI 2015 test set (200images)

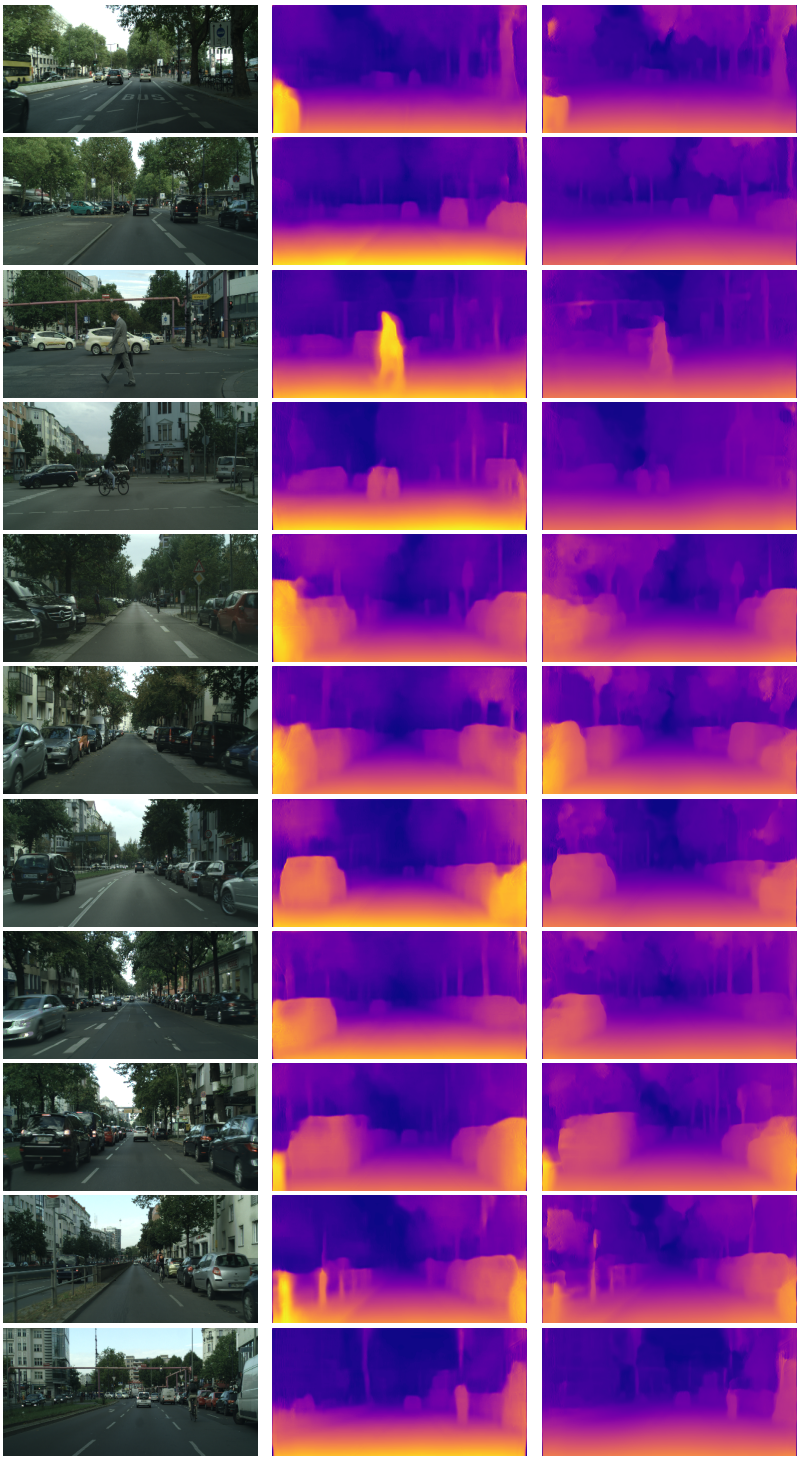
Method	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Original	**0.1217**	**1.011**	**5.221**	**0.218**	**0.852**	0.943	0.973
Competitive	0.1241	1.037	5.308	0.220	0.847	**0.943**	**0.973**

(b) Numerical Evaluation on the Eigen-split test set (697images)

Lower is better
Higher is better

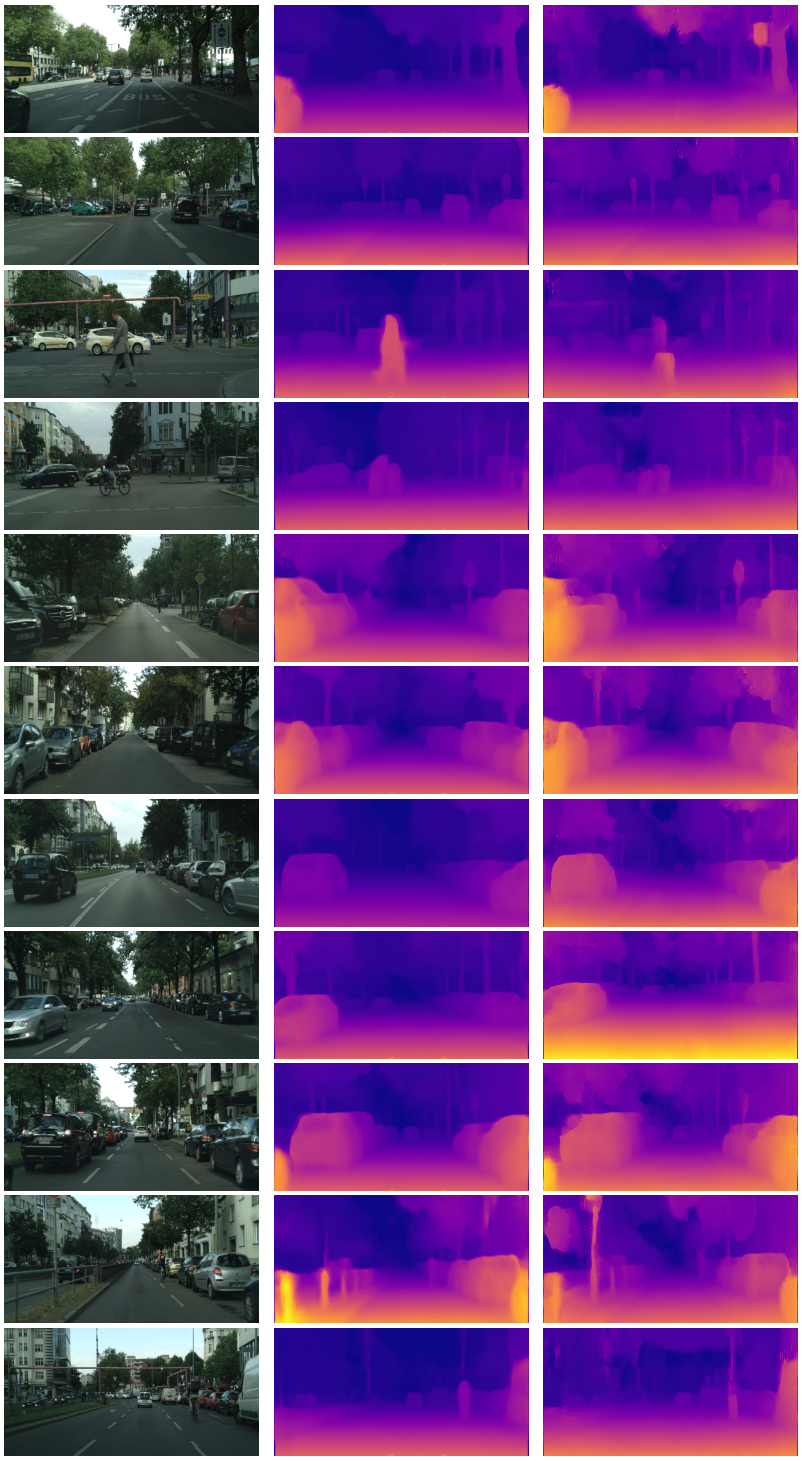
Table 4.9: Experiment 3: Numerical Evaluation

In the wake of the thorough evaluation of the proposed competitive training strategy, we take a further look at the performance of the resulting model and how it ranks among the best-performing state-of-the-art CNN-based methods for unsupervised depth estimation. We rely here on the reported results of the original best performing UMDLR method detailed in [6] and we compare the numbers produced by our competitive model against the available reported numbers.
The considered numerical test sets for evaluation are still the KITTI 2015 (200 images) and the testing Eigen-split set (697 test images). However for training, we choose to replicate the same training setup for the best-performing UMDLR from the paper. We use same data augmentation, the optimizer parameters and the training scheme that has been detailed in [6] for the sake of consistency.
A detailed summary of the numerical comparison of our competitive model's performance against state-of-the-art models is shown in Tab. 4.11. The first Tab. 4.11a contains the averaged results on the KITTI 2015 set where our model ranks first among all the selected models reported in [6]. It slightly improves upon the original best-performing UMDLR model trained with the same setup. This confirms the observation that considering semantic segmentation features during the training process using our competitive strategy boosts the quality of the produced unsupervised depth estimates. For the second test set



(a) Input frame (b) 2.Exp mono Model's result (c) 2.Exp competitive Model's result

Figure 4.9: The visual comparison between the competitive and the original model depth estimation on test set images of Cityscapes (2.Experiment).



(a) Input frame (b) 3.Exp mono Model's result (c) 3.Exp competitive Model's result

Figure 4.10: The visual comparison between the competitive and the original model depth estimation on test set images of Cityscapes (3.Experiment).

(Tab. 4.11b), the competitive model lands in second position among the other methods with a negligible advantage for the original UMDLR model in the case of 80m restricted depth range evaluation as well as for the 50m setup. Visual output samples of the best-performing models for the 80m case are presented in Fig. 4.11 which is courtesy of [6] except for our competitive model's results. The visual observation align with the numerical conclusions. Both original UMDLR and competitive UMDLR produce the best visually refined depth estimates against other methods with respect to the available ground-truth. That the competitive one has an advantage is not really clear to the bare eye comparison. However numerically, the averaging of all metrics over the whole test set (KITTI 2015) confirms this observation.

Method	dataset	Abs Rel	Sq Rel	RMSE	RMSE log	Log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
UMDELR with Deep3D	K	0.412	16.37	13.693	0.512	66.85	0.690	0.833	0.891
UMDELR with Deep3Ds	K	0.151	1.312	6.344	0.239	59.64	0.781	0.931	0.976
UMDELR No LR	K	0.123	1.417	6.315	0.220	30.318	0.841	0.937	0.973
UMDELR	K	0.124	1.388	6.125	0.217	30.272	0.841	0.936	0.975
UMDELR	CS	0.699	10.060	14.445	0.542	94.757	0.053	0.326	0.862
UMDELR	K+CS	0.104	1.070	5.417	0.188	25.523	0.875	0.956	0.983
UMDELR pp	K+CS	0.100	0.934	5.141	0.178	25.077	0.878	0.961	0.986
UMDELR ResNet pp	K+CS	0.097	0.896	5.093	0.176	23.811	0.879	**0.962**	0.986
Ours	K+CS	**0.095**	**0.889**	**5.007**	**0.165**	**23.737**	**0.883**	0.8971	**0.987**

(a) Numerical Evaluation on the KITTI 2015 test set (200images) trained on Cityscapes and fine-tuned using the whole KITTI split

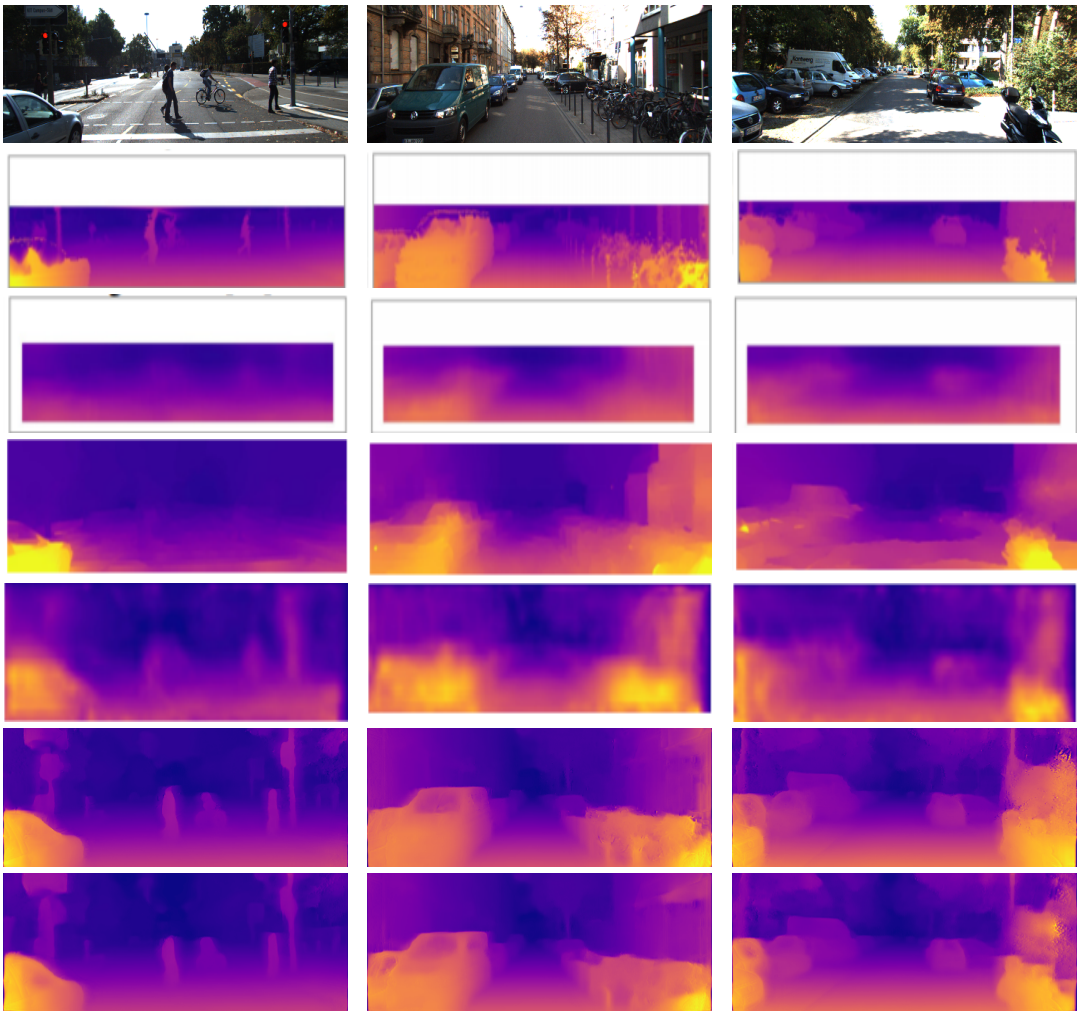
Method	dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Train set mean	K	0.361	4.826	8.102	0.377	0.638	0.804	0.894
Eigen *et al.* Coarse	K	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen *et al.* Fine	K	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu *et al.* DCNF-FCSP FT	K	0.201	1.584	6.471	0.273	0.68	0.898	0.967
UMDELR No LR	K	0.152	1.528	6.098	0.252	0.801	0.922	0.963
UMDELR	K	0.148	1.344	5.927	0.247	0.803	0.922	0.964
UMDELR	K+CS	0.124	1.076	5.311	0.219	0.847	0.942	0.973
UMDELR pp	K+CS	0.118	0.923	5.015	0.210	0.854	0.947	0.976
UMDELR ResNet pp	K+CS	**0.114**	0.898	**4.935**	**0.206**	0.861	0.949	**0.976**
Ours	K+CS	0.116	**0.895**	4.937	0.207	**0.861**	**0.950**	0.975
Garg *et al.* cap 50m	K	0.169	1.080	5.104	0.273	0.740	0.904	0.962
UMDELR cap 50m	K	0.140	0.976	4.471	0.232	0.818	0.931	0.969
UMDELR cap 50m	K+CS	0.117	0.762	3.972	0.206	0.860	0.948	0.976
UMDELR pp cap 50m	K+CS	0.112	0.680	3.810	0.198	0.866	0.953	0.979
UMDELR ResNet pp cap 50m	K+CS	**0.108**	**0.657**	3.729	**0.194**	0.873	**0.954**	0.978
Ours cap 50m	K+CS	0.112	0.659	**3.721**	0.195	**0.873**	0.950	**0.979**

(b) Numerical Evaluation on the Eigen-split test set (697images) training on Cityscapes ("Trainextra" + "Trainvaltest") and fine-tuned on the whole KITTI split

Lower is better

Higher is better

Table 4.11: State of the art Comparison



(a) Scene 1 (b) Scene 2 (c) Scene 3

Figure 4.11: The visual comparison of state-of-the art methods on images from the KITTI test Eigen split (courtesy of [6]). The ground truth velodyne depth interpolated because of sparsity. From top to bottom: Input frame, interpolated ground truth, Eigen *et al.* result, Liu *et al.* result, Garg *et al.* result, UMDELR ResNet pp result, Ours (competitive).

5 Conclusions

In the proposed work, a generic enhancement of urban scene understanding for autonomous driving scenarios was targeted. Under precise requirements pre-defined within the *Audio and Video Advanced Driving Assistance Systems* (AVADAS) project of HUAWEI, exclusive priority to emerging DL technologies was accorded. Surfing on the current wave, this research work presents an alternative to tackle the problem of improving scene understanding using CNNs. From the visual nature of the understanding emanates the high priority given to the camera sensors. As defined by the mentioned project requirements, cameras were considered as the only source of information to the autonomous vehicle within this work. Therefore, the motivation was not to present an umpteenth algorithmic solution to classical computer vision problems but rather proposing a new DL-based technology to tackle these issues from a different point of view. The final goal is to improve upon state-of-the-art CNN-based methods that already overshadowed the classical landscape.

In this context, we start by solving the problem of semantic segmentation using CNNs. We see this as the starting point for improving scene understanding faculties of autonomous systems. Semantic segmentation gives the vehicle a proper perception of the incoming visual information as independent semantically meaningful classes. This guarantees an improved analysis and perception of the navigation environment leading subsequently to better driving decisions. For this, a class-level CNN-based semantic segmentation solution was proposed under the name of MokaNet. The model is characterized by a high-complexity structure with densely connected sequential blocks. Each of which is designed as a concatenation of different sub-blocks embracing an encoder-decoder layout with long-range links joining the two parts. The newly proposed connection strategy, the cascaded redundant architecture and the incorporation of additional specific layers (Batch-normalization, flat convolution layers, etc...) are the principal highlights of the MokaNet architecture adapted to learning semantic segmentation. Moreover, the application of the model on complex tasks related to the segmentation context such as car instance separation proved the efficiency and robustness of not only the model itself but also the used training scheme including data augmentation, model design and parameter choice. The extended semantic segmentation solution with instance-level car separation was delivered to meet the project requirements by giving the highest priority to the car object-class. The motivation behind this is indeed to anticipate processing steps in later stages of the project where filtering out dynamic cars is needed for optimizing the localization step of the SLAM approach.

A detailed evaluation of the different aspects of the proposed approach including robust-

ness, generalization capacities, visual performance and numerical accuracy has been conducted. Additionally, a thorough state-of-the-art comparison against publicly available CNN-models of the Cityscapes benchmark was performed. Exclusively, models trained to solve the semantic segmentation task within autonomous driving context were considered. MokaNet delivered excellent visual and numerical results. Despite limited training data size, it proved to be robust enough against extreme challenging content variations of the test samples (e.g. datasets depicting traffic scenes from various countries) and significantly improved upon state-of-the-art methods.

After tackling the semantic segmentation problem, an extensive study of its effect upon other simultaneously learned tasks was realized. As a main example, supervised disparity estimation within a multi-tasking CNN-model trained with semantic segmentation was considered. First, a personally developed approach based on available segmentation labels was used to generate ground-truth masks for disparity planes estimation. These masks encode uniform displacement information of semantic segments between left and right image views. The resulting ground-truth combining segmentation and disparity information was harnessed to challenge the ability of the proposed model to decipher high-complexity mappings. This has been examined in the case of basic and cooperative bi-decoder architectures (consisting of two decoding parts one for segmentation and the other for disparity planes estimation). Subsequently, we evaluated the effect of jointly optimizing tightly related tasks within multi-objective optimization frameworks with a specific focus on the effect of high-quality segmentation upon enhancing the estimation of disparity planes.

An extension of the same experiment to pixel-wise depth estimation was later realized. Unlike the previous case, results didn't show any significant improvement of the jointly trained model (Multi-tasking model with supervised depth and segmentation simultaneously learned) upon the basic DispNet (only supervised depth estimation) in terms of estimating pixel-wise supervised depth from stereo images. In fact, the lack of reliable real-world depth ground-truth appeared to be a determinant failure factor. The training procedure was clearly disturbed by the use of artificially rendered images instead of real data.

In order to overcome the limited data issue, we shifted the focus towards the unsupervised depth estimation approach that only requires stereo pairs without need of available ground-truth for training. The best-performing unsupervised model for depth estimation called UMDLR was considered. Although experiments confirmed the fact that learning semantic segmentation following the standard cooperative multi-tasking strategy could significantly improve the unsupervised learning of depth, the restricted size of the available datasets necessary to perform such training procedures posed again a big challenge. For this, we proposed a new conditional strategy for training multi-tasking CNNs inspired by game theory. With our new strategy, training such models becomes possible using independent heterogeneous datasets relevant for different tasks instead of requiring a single dataset with all ground-truth labels available at once. It is based on performing the optimization of these tasks in an alternating competitive way instead of linearizing their objective functions. Our solution outperforms state-of-the-art approaches and competes on high levels

with the standard cooperative multi-tasking training. By picking up data from heterogeneous datasets in every iteration, our training method gives much more flexibility to the CNN designer. At the same time, it totally profits from the transfer learning between semantic segmentation and other applications as confirmed by the experiments conducted in Chapter 4 for unsupervised depth estimation. In fact, our approach delivers a comparable performance to the one achieved through cooperative multi-tasking without being limited by data availability.

Bibliography

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[2] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, 2017.

[3] M. P. Shah. (2017, Jan.) Semantic segmentation architectures implemented in pytorch. [Online]. Available: https://github.com/meetshah1995/pytorch-semseg

[4] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[5] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 06 2015.

[6] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[7] Association For Safe International Road Travel. (2020, Jan.) Annual global road crash statistics. [Online]. Available: https://www.asirt.org/safe-travel/road-safety-facts/

[8] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey. dot hs 812 115," *National Highway Traffic Safety Administration, US Department of Transportation*, 2015.

[9] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous Driving: Technical, Legal and Social Aspects*. Springer, 2016.

[10] L. Fei-Fei, A. Iyer, C. Koch, and P. Perona, "What do we perceive in a glance of a real-world scene?" *Journal of vision*, vol. 7, no. 1, pp. 10–10, 2007.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[12] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[13] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic imaging*, vol. 13, no. 1, pp. 146–168, 2004.

[14] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[15] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: An evaluation of the state-of-the-art," *Computer Vision and Image Understanding*, vol. 166, pp. 1–27, 2018.

[16] G. Mihai, L. Stanescu, D. Burdescu, A. Doringa, M. Brezovan, and E. Ganea, "A comparison of three graph-based image segmentation algorithms." pp. 480–485, 01 2010.

[17] B. Fulkerson and S. Soatto, "Really quick shift: Image segmentation on a gpu," in *European Conference on Computer Vision*. Springer, 2010, pp. 350–358.

[18] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 583–598, 1991.

[19] Qure.ai. (2020, Jan.) semantic-segmentation-deep-learning-review. [Online]. Available: http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review

[20] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[24] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

[25] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[26] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," in *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017, pp. 1–4.

[27] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[28] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] M. A. Islam, M. Rochan, N. D. Bruce, and Y. Wang, "Gated feedback refinement network for dense image labeling," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4877–4885.

[30] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," in *Advances in neural information processing systems*, 2015, pp. 1495–1503.

[31] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*, vol. abs/1412.7062, 2014. [Online]. Available: http://arxiv.org/abs/1412.7062

[32] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018. [Online]. Available: https://doi.org/10.1109/TPAMI.2017.2699184

[33] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision (ICCV)*. IEEE, 2017, pp. 2961–2969.

[36] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 924–933.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2014, pp. 580–587.

[38] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5696–5704.

[39] J. van den Brand, M. Ochs, and R. Mester, "Instance-level segmentation of vehicles by deep contours," in *Asian Conference on Computer Vision (ACCV)*. Springer, 2016, pp. 477–492.

[40] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "Instancecut: from edges to instances with multicut," in *30th IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, pp. 7322–7331.

[41] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "Sgn: Sequential grouping networks for instance segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3496–3504.

[42] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 441–450.

[43] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 293–301.

[44] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *German Conference on Pattern Recognition*. Springer, 2016, pp. 14–25.

[45] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[46] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 2858–2866.

[47] Z. Zhang, S. Fidler, and R. Urtasun, "Instance-level segmentation for autonomous driving with deep densely connected mrfs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 669–677.

[48] I. Halfaoui, F. Bouzaraa, and O. Urfalioglu, "Cnn-based initial background estimation," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 101–106.

[49] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[50] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, vol. 00, Oct. 2018, pp. 5000–5009. [Online]. Available: doi.ieeecomputersociety.org/10.1109/ICCV.2017.534

[51] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[54] Cityscapes Team. (2020, Jan.) Cityscapes benchmark suite. [Online]. Available: https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the european conference on computer vision*. Springer, 2014, pp. 740–755.

[56] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 297–312.

[57] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 859–868.

[58] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[59] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, "Analyzing modular cnn architectures for joint depth prediction and semantic segmentation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4620–4627.

[60] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[61] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2800–2809.

[62] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, "Joint semantic segmentation and 3d reconstruction from monocular video," in *European Conference on Computer Vision*. Springer, 2014, pp. 703–718.

[63] A. Mousavian, H. Pirsiavash, and J. Košecká, "Joint semantic segmentation and depth estimation with deep convolutional networks," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 611–619.

[64] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 213–228.

[65] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *European Conference on Computer Vision*. Springer, 2014, pp. 756–771.

[66] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[67] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[68] A. Saxena, J. Schulte, A. Y. Ng *et al.*, "Depth estimation using monocular and stereo cues." in *Proceedings of the international Joint Conference on Artificial Intelligence (IJCAI)*, vol. 7, 2007, pp. 162–170.

[69] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[70] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

[71] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[72] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[73] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, Jan. 2020. [Online]. Available: http://www.blender.org

[74] R. Stoop, J. Buchli, G. Keller, and W.-H. Steeb, "Stochastic resonance in pattern recognition by a holographic neuron model," *Physical Review E*, vol. 67, no. 6, p. 061918, 2003.

[75] I. Halfaoui and O. Urfalioglu, "Improving bm3d on non-stationary gaussian models for real image noise," in *Proceedings of the international Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE, 2015, pp. 467–472.

[76] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 842–857.

[77] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 6, 2017, pp. 1851–1858.

[78] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[79] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6647–6655.

[80] H. Jiang, G. Larsson, M. Maire Greg Shakhnarovich, and E. Learned-Miller, "Self-supervised relative depth learning for urban scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–35.

[81] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[82] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[83] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[84] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, and A. M. López, "Monocular depth estimation by learning from heterogeneous datasets," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 2176–2181.

[85] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[86] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[87] J.-A. Désidéri, "Split of Territories in Concurrent Optimization," INRIA, Research Report RR-6108, 2007. [Online]. Available: https://hal.inria.fr/inria-00127194

[88] A. Minelli, I. Salah El Din, G. Carrier, A. Zerbinati, and J.-A. Désidéri, "Cooperation and Competition Strategies in Multi-objective Shape Optimization - Application to Low-boom/Low-drag Supersonic Business Jet," in *43rd AIAA Fluid Dynamics Conference and Exhibit*. San Diego, California, United States: The American Institute of Aeronautics and Astronautics (AIAA), Jun. 2013. [Online]. Available: https://hal.inria.fr/hal-00935320