Technische Universität München
TUM School of Computation, Information and Technology

Domain Knowledge Exploitation for Machine Learning Applications in the Context of Automated Driving and Manufacturing

Matthias Arnold Pollach, M.Sc.

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Nassir Navab

Prüfende der Dissertation:

1. Prof. Dr.-Ing. habil. Alois Knoll
2. Prof. Dr. Binh Q. Tran

Die Dissertation wurde am 07.09.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 11.04.2024 angenommen.

Abstract

The challenge of using machine learning at scale has existed since the 1950s, with limited adoption until recent advances in computational resources improved artificial neural networks' performance.

Currently, machine learning is mainly used in specific areas like consumer products, with limited industry-wide adoption. Historically, it relied on knowledge engineering for contextual information, but deep learning eliminated the need for manually created knowledge bases. Increased computational resources and large datasets improved image recognition, leading to high demand for annotated training data, feasible for some applications but prohibitive for most industrial uses. The complexity of deep neural networks poses significant challenges, requiring substantial effort to maintain and making debugging difficult.

Extensive retraining is needed when use cases or system components change, increasing the need for matching training data. Machine learning has shifted from a computational resource problem to a data problem.

This work proposes novel approaches to exploit human expert domain knowledge to reduce training needs and simplify machine learning architectures. Preserving information from multi-modal sensor data is critical for object detection and perception. Two sensor fusion approaches leverage existing domain knowledge: one reduces complexity to determine fusion network architecture, and the other uses knowledge of object appearance, leveraging object detection datasets for semantic segmentation tasks.

A new framework for creating multi-stage perception systems combines traditional and deep learning approaches. A knowledge graph determines the cascade of classifiers, making models more explainable and allowing retraining of individual components. This architecture allows for adaptive classification depth, managing computational resources efficiently.

Lastly, an approach to tailoring neural networks to industrial use cases based on domain knowledge is demonstrated for various industry applications.

The concepts introduced provide a foundation for exploiting domain knowledge to enable more machine learning-driven applications. Reduced training data needs offer a major advantage over current deep learning systems, reducing implementation effort.

Zusammenfassung

Die Herausforderung, maschinelles Lernen in großem Maßstab einzusetzen, besteht seit den 1950er Jahren. In den letzten zehn Jahren haben exponentiell wachsende Rechenkapazitäten zu bedeutenden Fortschritten geführt, insbesondere in der Bilderkennung. Allerdings wird maschinelles Lernen bisher nur in bestimmten Bereichen wie Endkundenprodukten eingesetzt und ist noch nicht branchenübergreifend verbreitet.

Früher basierte maschinelles Lernen auf Wissensmodellierung und expertengesteuerter Merkmalskonstruktion. Methoden des tiefen Lernens haben die manuelle Erstellung von Wissensdatenbanken überflüssig gemacht und die Bedeutung der Merkmalskonstruktion verringert. Größere Rechenressourcen und große Datensätze haben zu einer verbesserten Leistung geführt, was jedoch eine hohe Nachfrage nach annotierten Trainingsdaten zur Folge hat, die oft unwirtschaftlich ist.

Die Komplexität tiefer neuronaler Netze stellt zusätzliche Herausforderungen dar. Ihre Wartung erfordert erheblichen Aufwand, und ihre Ergebnisse sind schwer zu erklären. änderungen oder Weiterentwicklungen erfordern erneutes Training und passende Trainingsdaten. So hat sich maschinelles Lernen von einem Rechenressourcen- zu einem Datenproblem entwickelt.

Diese Arbeit schlägt neuartige Ansätze und ein Rahmenwerk vor, um den Trainingsbedarf durch menschliches Fachwissen zu verringern und die architektonische Komplexität zu reduzieren. Multimodale Sensordaten sind für die Objekterkennung entscheidend, da sie einen vielfältigeren Merkmalsraum bieten. Zwei Ansätze für die Sensorfusion in einem frühen Stadium werden vorgestellt: Einer nutzt Expertenwissen zur Reduzierung der Komplexität und Bestimmung der Netzwerkarchitektur, der andere basiert auf Wissen über das Aussehen von Objekten. Letzterer Ansatz ermöglicht auch die Nutzung von Objekterkennungsdatensätzen für semantische Segmentierungsaufgaben.

Ein neues Rahmenwerk für mehrstufige Wahrnehmungssysteme wird präsentiert, das traditionelles maschinelles Lernen und tiefe Lernansätze kombiniert. Ein Wissensgraph steuert die Kaskade von Klassifikatoren, die flacher und weniger komplex als tiefe Lernansätze sind. Dies führt zu besser erklärbaren Modellen und ermöglicht es, einzelne Komponenten neu zu trainieren. Zudem erlaubt diese Architektur eine situationsadaptive Klassifizierungstiefe zur aktiven Verwaltung von Rechenressourcen.

Schließlich wird ein Ansatz zur Anpassung neuronaler Netze an industrielle Anwendungen basierend auf explizitem Domänenwissen vorgestellt und demonstriert. Die Ergebnisse und Konzepte dieser Arbeit bilden die Grundlage für die Nutzung von Domänenwissen, um weitere Anwendungen des maschinellen Lernens zu ermöglichen, was den Bedarf an Trainingsdaten und den Implementierungsaufwand verringert.

Contents

1 Introduction

Can machine learning and artificial intelligence be applied to almost all desired applications at scale without requiring an unfeasible increase in available training data and computational resources?

Artificial Intelligence (AI) is not a new domain, it has been used in the last century, and the first mention of an AI system dates back to 1955 [1]. Since there have been multiple waves of Machine Learning (ML) and AI, which had not led to a breakthrough use on a large scale, each wave sparked great interest in the domain due to promising results but constraints and limitations prohibited a wider use of the proposed approaches.

Systems in the 60s and 70s relied on knowledge engineering concepts. They required significant human input to create knowledge bases, which enabled early inference engines. Real-world problems rely on humans to provide contextual knowledge. This is not feasible for complex issues because symbolic representation and the absence of a mathematical problem representation create a not impermeable bottleneck to create an actionable knowledge representation.

In the 80s, Multi-Layer Perceptron (MLP)s were introduced, and superior ML systems evolved. They did not rely on a manually crafted knowledge base because the inference was generic and did not rely on human input. The model is trained based on examples, and a loss function is minimized during training. This results in a generic component, which can be used for unknown input during inference by computing features. Identifying features that allow efficient separation of the feature space requires domain expertise and was a blocker for using such systems at scale.

Over the last decade, deep learning has evolved and led to a breakthrough in applying ML at scale for various applications. The underlying theoretical foundation has remained the same, but the computational resources enabled the use of Artifical Neural Network (ANN)s in an unprecedented way. Stacking many layers of neurons on top of each other resulted in powerful feature spaces, which allowed for a much better separation of classes. The breakthrough of deep learning started with outperforming all other approaches in the ImageNet Large Scale Visual Recognition Challenge 2012 [2] by implementing a deep Convolutional Neural Network (CNN). Manually created knowledge bases and feature engineering were no longer required for this approach because the feature space representations and their separations are learned during training. This happens based on annotated training data. However, this has resulted in a trend where the demand for training data is constantly increasing, which is feasible for some applications but prohibitively complex and expensive for others. Data availability has become a blocker for many use cases and prevents the utilization of ML for numerous domains where great value could be generated.

1.1 Limitations of Deep Learning

The current state-of-the-art Deep Neural Network (DNN)s are trained end-to-end using annotated training data. In general, the deeper and wider a network becomes, the more trainable parameters there are, and the more training data is required. This approach works well for frequently occurring examples with readily available training data. However, in case multi-modal sensor input is used for a Neural Network (NN), changing a sensor poses a problem because the entire network needs to be retrained based on data from the new sensor. This requires aligned sensor data from all sensors, which has to be annotated. In many industrial applications where the generation of multi-modal sensor data is expensive, this presents itself as a blocker for a broader application of DNNs. The used architectures for NNs are not modular in the sense that only a subset of the network could be retrained. Retraining a multi-model NN with a single sensor modality would result in catastrophic forgetting, which means the network would not work as desired when using multi-modal input during inference. This is caused by adapting the weights of the NN purely to the presented data.

Despite these shortcomings, DNNs have been widely used for consumer applications related to image or natural language processing, available on modern smartphones. However, for these use cases, the requirements vary vastly from those found in industrial applications. In particular, the cost of wrong classifications is very different. For consumer applications, an error caused by a NN might result in a degraded user experience. In contrast, the consequences in industrial applications can be costly, and in the case of safety-critical applications like Autonomous Driving (AD), such errors might even be fatal [3].

In the context of AD and other robotic or industrial applications, it is especially crucial to ensure safe and reliable operations. The complexity of DNNs poses a significant challenge to reasoning about the produced outcome and prohibits formal validation approaches. Adversarial attacks on NNs are another danger to safety-critical applications. Small changes in the input signal, which a human might not perceive, can result in drastic changes in the output.

Explainable AI has evolved as a field of research to provide more insights into DNNs. Various visualizations to understand which neurons fire for a given input have been introduced, as well as heatmap-based approaches, which show the importance of image areas for the result. Some techniques allow us to trace the prediction result back to the input to understand the working principle better. However, such methods provide more insights into DNNs, which helps improve feature engineering or architectures. This will help enable additional use cases and enhance existing applications, but the problem remains that DNNs are black-box classifiers. Explainable visualization tools do not allow us to overcome the limitations of NNs, especially when safety-critical decisions depend on the output [4, 5, 6].

1.2 Introduction to Domain Knowledge Exploitation

The ML problem has transitioned into a data problem. The need for increased training samples to enable NNs, combined with the known shortcomings of deep learning, has prevented their use for various applications. The idea of knowledge engineering has been replaced by leveraging data. The result is NNs that learn contextual knowledge, even though it would be readily available.

The approach of domain knowledge exploitation for ML use cases takes advantage of knowledge modeling. This knowledge is explicitly incorporated into the system, eliminating the need for a DNN to implicitly learn constraints and boundary conditions based on training data. Instead of increasing the learning complexity of a DNN by adding more neurons and layers, the domain knowledge can directly be exploited. This leads to a smaller neural network, reducing the amount of training data that must be collected to obtain the desired results [7].

The selection of training data itself significantly impacts the resulting NN and its quality. The variety and annotation quality strongly contributes to the training outcome. Domain knowledge is essential for selecting and annotating training samples in such a way that a network can be appropriately trained [8].

A ML algorithm's effectiveness is determined by the training data and the amount of domain knowledge that goes into selecting training data and designing network architectures. The relationship between training data and expert domain knowledge exploitation shows the power of this approach. The limitations of deep learning concerning extensive training datasets are especially counteracted. Explicit knowledge modeling enables the design of systems, which are explainable at different stages if they are set up in a modular way and follow the knowledge representation [9].

Overcoming and addressing some of the existing limitations of deep learning through domain knowledge exploitation enables the refinement of existing ML use cases while providing the opportunity to apply ML and AI to new use cases. Existing and new use cases pose domain-specific challenges, introduced in the following and addressed throughout this thesis by exploiting domain knowledge.

1.3 Automated and Autonomous Driving

Every year more than 1.25 million people die in traffic accidents, and roughly 20 million are seriously injured [10]. Human failure is the leading cause of traffic accidents, accounting for more than 90% of incidents [11]. Over the last decade, politics and industry have invested a significant effort to reduce the number of accidents. One outcome is Advanced Driver-Assistance Systems (ADAS), which helps decrease the number and the severity of accidents by alerting and supporting the driver.

The next step in the evolution of ADAS and mobility, in general, is already taking place and is increasing the level of driving automation. The ultimate goal of this process

is the implementation of fully AD.

The main challenge of reaching AD is to perceive the environment. The car must be able to judge the current driving situation at all times, which is comparable to the task a human driver has to accomplish. One of the keys to achieving this goal is detecting the objects proximate to the vehicle. This has to be done as reliably as possible to guarantee a safe operation. This can be achieved by fusing data from different sensors and different sensor modalities. This work uses Light Detection and Ranging (LiDAR) sensors, cameras, and Radio Detection and Ranging (RADAR) sensors.

There are various approaches to sensor fusion. One popular method is to combine the output of so-called smart sensors. Such sensors have major computational capabilities, and signal processing is included to detect objects. This is state-of-the-art in currently available and upcoming ADAS systems. Nevertheless, processing all available information to increase detection confidence while reducing latency is essential. The time delay until an object is perceived by an AD system is of great interest, as it is one of the most safety-critical aspects of AD. The use of smart sensors cannot guarantee that all captured information is used for the driving decision because each sensor filters data without taking information from other sensors into account.

Smart sensors usually rely on object tracking to detect objects and to increase object detection confidence. This negatively impacts their detection latency but improves the handling of partial or entire object occlusions. Directly coupling the sensors to the sensor fusion system reduces the latency and avoids filtering raw sensor data before the fusion step. The low-level sensor fusion, operating on associated raw sensor data, increases the object detection confidence. Fusing multiple sensor modalities helps overcome the specific weaknesses of individual sensor technologies.

Motivation

In the last decade, driver assistance systems have become very popular and common. Such systems increase the level of comfort and safety. ADAS systems are state-of-the-art in most cars sold today. In the future, there will be highly automated and potentially fully AD. An overview of the expected development timeline can be found in [12]. Depending on the application and the complexity of the driving task, an assistance system takes control of different functions. The Society of Automotive Engineers (SAE) levels describe the level of driving automation, according to figure 1.1.

Early ADAS systems require full driver attention at all times. This section gives an overview of the levels of automation. Examples of ADAS systems are provided to illustrate these levels. Further details can be found in [12].

- Lane change assist
- Lane departure warning
- Front collision warning
- Emergency braking assist

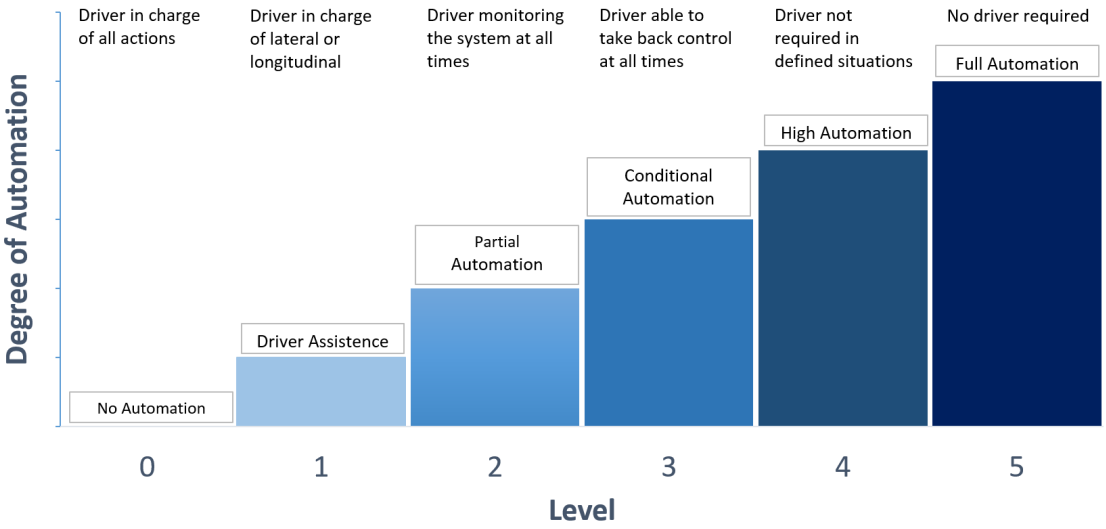


Figure 1.1: Levels of Automation

- Park Distance Control (PDC)
- Adaptive Cruise Control (ACC)

A PDC system gives visual and acoustical hints to the driver to simplify parking maneuvers. In general, level 0 autonomy systems only give hints and warning signals to the driver but do not take control of any actuator. Front collision warning, lane departure warning, or PDC are examples of level 0 systems. In contrast, level 1 assistance systems take partial control over actuators. The driver has to monitor the system and simultaneously operate the actuators, which are not controlled by the car. An example of a level 1 autonomy system is an emergency braking assistant. Suppose a vehicle, a pedestrian, or any other object is detected in front of the ego vehicle, and a collision could occur. In that case, the system triggers an emergency braking maneuver to avoid an accident or reduce the damage.

Level 2 autonomy systems introduce more comfort and increase safety. An example is a traffic jam assistant with included lane keeping and ACC. The driver must constantly monitor the system, even though the car can take over lateral and longitudinal control.

Any system level 3 or higher is considered an automated driving system. In contrast to level 2, a system of level 3 does not need permanent driver supervision. The vehicle has control over the actuators in well-defined situations. The driver must take over within a specific time window in an unknown situation. This leads to the fact that safe operation strongly depends on the ability of the driver to take back control of the car. Such a control handover is automatically triggered whenever the system performance is insufficient.

In defined driving situations, a level 4 system can drive autonomously. A safety fallback is triggered in case of unknown situations or the case that the vehicle is about

to leave the defined driving situation and whenever the driver is not taking control when asked to do so. The system is able to bring the car to a safe state. A level 5 car is considered a fully autonomous vehicle that can handle all situations. Human supervision is not required.

The higher the level of automation is, the stricter the safety requirements are. Vehicles have to be able to perceive the environment reliably at all times. However, the environment permanently changes, and, like humans, autonomous vehicles have to detect objects in the surroundings to provide a high level of automation while operating safely. Early driver assistance systems rely on a single sensor modality or even a single sensor. With an increasing level of automation, sensor redundancy is necessary to increase system safety and performance. A sensor data fusion approach is required to properly handle the information from different sensors. Ensuring higher levels of automation relies heavily on the selection of appropriate sensors and sensor fusion.

Sensor Selection and Setup for Automated Driving Applications

The choice of sensor technologies, as well as the total number of sensors and their setup, is crucial for the success of any driver assistance system. Multiple sensors capture more data, and therefore, additional information is provided, which has to be analyzed. To guarantee a reliable perception of the environment under all environmental conditions, a variety of different sensors and sensor modalities have to be used. Depending on environmental conditions, different sensor technologies show various strengths and weaknesses. Object detection systems benefit from multiple sensors, multiple modalities, and different viewing angles of the same object. The chosen sensor technologies used for this present work are explained in this chapter. Additionally, the Field of View (FOV) of neighboring sensors needs to overlap when selecting a sensor setup. A possible configuration and a mapping of ADAS functions and sensors are shown in figure 1.2.

The area in front of the car is especially very safety-critical, and objects must be detected with a high level of confidence. The number of overlapping FOVs is high, and redundancy is introduced. The sensor fusion system benefits from this, and the confidence level for object detection is increased. In the following, the three most crucial sensor technologies are briefly described.

LiDAR

There is a variety of different LiDAR sensors on the market. All LiDAR sensors are based on the principle of emitting light pulses and measuring the time of flight until reflections arrive. Most currently available automotive-grade LiDARs are mechanical scanning ones. The number of scanning layers can vary from one to 64. Depending on the semiconductor material, the wavelength is either 905 nm for silicon or around 1550 nm for gallium arsenide [14]. In the future, solid-state LiDARs are expected to reduce sensor costs and increase robustness, as they do not have any moving parts. The information the LiDAR sensor captures is a point cloud. Each point has an x, y, and z coordinate. Additional information can be included, for example, intensity values. The

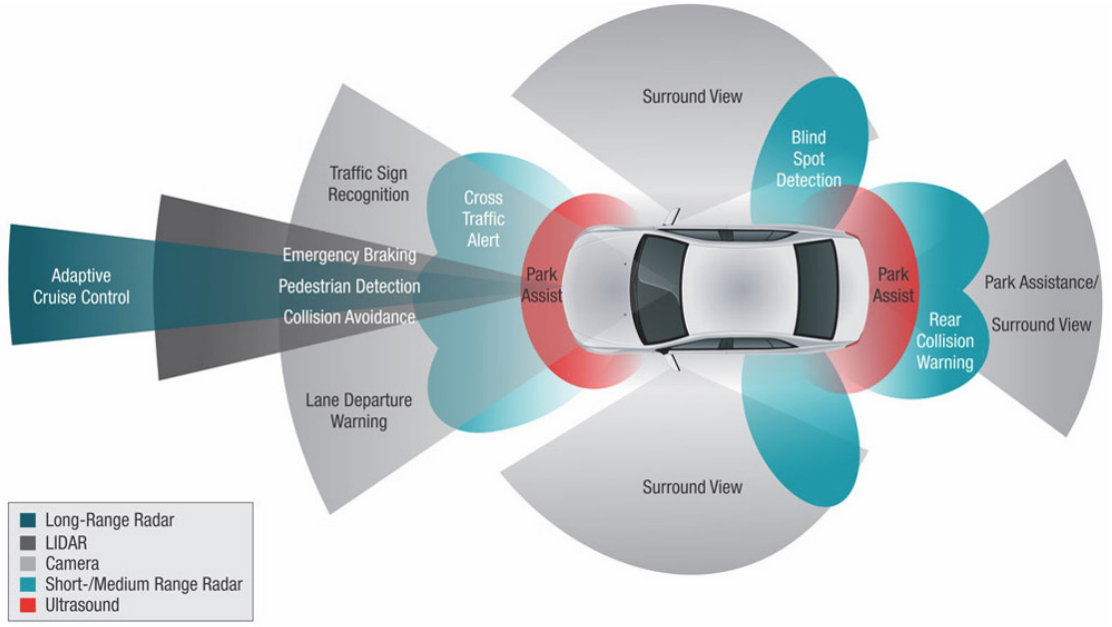


Figure 1.2: This figure shows a possible sensor setup and ADAS functions and and are from [13]

benefit of using LiDARs is the high measurement density compared to other sensors, like RADAR. Objects proximate to the vehicle have a higher measurement density than objects further away. Detection confidence increases as more points are associated with an object, as demonstrated in chapter 4. Data captured by LiDAR is very reliable, and this property is beneficial, especially in safety-critical applications. For example, an emergency braking assistant must have reliable information about the objects in front of the vehicle. LiDAR information is well-suited for such a purpose and can be used to improve object detection.

Nevertheless, all those advantageous properties are available when the environmental conditions are not negatively impacting sensor data capturing. Fog, rain, and snow reduce the performance and reliability of a LiDAR sensor. Furthermore, the scanning rate of a LiDAR is lower in comparison to other sensors.

RADAR

A sensor that has already been widely used is the RADAR sensor. One has to differentiate between short-range RADARs operating at 24 GHz and long-range RADARs operating at 77 or 79 GHz. Most RADARs used in automotive applications are Frequency-Modulated Continuous Wave (FMCW) RADARs [15]. The captured data usually includes the distance, Signal-to-Noise Ratio (SNR), radial velocity, and azimuth angle of a detected target. Some RADARs are able to return an elevation angle measurement.

A large number of modern cars are already equipped with RADAR sensors. They are being used, for example, for emergency braking or ACC systems. In the context of this present work, the RADAR measurements are helpful because they provide instant information about the velocity of an object. Another benefit of RADAR sensors is the availability of intensity values, which can be used to determine the properties of an object, for example, by using the RADAR cross-section as an indicator.

A disadvantage of RADARs is the limited resolution compared to LiDAR sensors. High spatial resolution is essential for object detection in urban use cases. Furthermore, the measurements suffer from noise. This can be resolved by fusing the RADAR measurements with data from other sources.

Camera

A widely spread sensor in modern vehicles is the camera. There is a large variety ranging from grayscale to color, low-resolution to HD cameras, and rolling to global shutter. Depending on the application, the difference between rolling and global shutter can be significant. The rolling shutter has some significant disadvantages due to motion distortion. The output of a camera is an image consisting of pixels. Many cars use cameras for several applications, such as lane detection, emergency braking, or parking assistance.

A significant advantage of images is that humans can easily understand the output and that there are a lot of well-known image processing techniques. This is beneficial for object detection. For example, some approaches use neural networks for image-based object detection.

Using cameras has several drawbacks. Like the human eye, cameras suffer, for example, from bright light, fog/smog, rain, or snow. Those effects reduce the visibility and, consequently, the information provided by the camera. Another issue with multiple cameras is that high data rates must be processed in real-time. This means that enough computational resources must be provided, especially for embedded implementations, and this is a challenge.

In addition to the mentioned sensor modalities, a large variety of other sensors are needed for automated driving. Vehicles are usually equipped with ultrasonic sensors and a Global Positioning System (GPS).

As mentioned, the higher the level of automation, the more critical it is to have a reliable perception of the environment. This is achieved by using multiple sensors. Therefore, combining information from various sensors is necessary to overcome this issue.

Challenges

The goal of every sensor fusion approach is to perceive the environment and objects within it as accurately as possible. Various challenges must be met to do that, which are very similar for all sensor fusion approaches. Every type of sensor has different strengths and weaknesses, which have to be taken into account. A measurement from a

sensor does not necessarily relate to a real object and is therefore referred to as a False Positive (FP). On the other hand, it is not guaranteed that every object corresponds to a measurement. This is generally referred to as a False Negative (FN). Increasing the quality of an object detection system requires reducing the number of FPs and FNs. In the following, a selection of the most relevant challenges is described:

- **Temporal synchronization:** Different sensor modalities are usually not synchronized and run at a frequency of 10 - 100 Hz. This can vary; for example, a mechanical LiDAR sensor is running at a constant frequency, whereas a camera might change its frequency. In addition, most sensors introduce latency due to pre-processing and data transfer. Data from all sensors must be fused to get an optimal result regarding the existence likelihood. Before that, the data has to be associated, which requires suitable synchronization strategies. An overview of the state-of-the-art synchronization approaches can be found in [12] and [16].

- **Spatial synchronization:** Due to varying frequencies and a temporal offset among sensors, a spatial offset is introduced. The data of a fast sensor has to be associated with the data of a slower one. The larger the time difference is, the larger the spatial offset. Consequently, a suitable spatial synchronization technique is mandatory for a reliable association, which is crucial for a sensor fusion system. An overview of the state-of-the-art solutions is given in chapter 3, and a synchronization method is proposed.

- **Measurement noise:** Every measurement system suffers from random fluctuations in the measured signal and undesired and inaccurate measurements. Statistical parameters and suitable noise models can partly model this. A single RADAR measurement, referred to as RADAR target, is a suitable example. The position is given by three coordinates (x,y,z). Due to noise, the position can also be described as a Gaussian distribution with a variance directly proportional to the noise intensity. The influence of noise depends on many conditions, like temperature and air humidity. For example, a LiDAR sensor is very sensitive to fog [17]. As a result, the measurement quality drastically varies compared to situations without fog.

- **FOV and handover:** Handing targets from one sensor over to another is essential. Each sensor has a FOV, and only within that FOV measurements are expected. Depending on the sensor modality, the data quality varies within the FOV, which has to be considered using suitable sensor models. According to the ISO26262 norm [18], an autonomous vehicle has to be equipped with a sensor setup so that every spot around the car is monitored, which results in sensor redundancy. In such a setup, overlapping FOVs of multiple sensors are present. A sensor fusion system targeting the minimization of detection latency has to hand over targets from one FOV to another across different sensor modalities. Without this capability, additional processing effort and latency are introduced. For example, a RADAR system covering 360 degrees of vehicle surroundings has multiple sensors and overlapping FOVs. An object moving from one FOV to another has

to be monitored. A high-level fusion concept requires the sensor to observe an object repeatedly and match tracks to ensure a successful handover. This additional latency can be avoided by using a low-level fusion concept.

- **Ego-motion compensation:** Another essential requirement for robust object detection is ego-motion estimation. Accurate information about ego-motion is necessary to distinguish the motion of other objects, like vulnerable road users, from ego-motion. Consequently, a suitable compensation method must be used to perceive the environment and determine accurate state information reliably. An overview of possible methods can be found in [19].

- **Calibration:** Calibration of all sensors is another challenge that has to be solved. If the sensors are not calibrated properly, the sensor data will not be spatially aligned and will have an offset. This negatively impacts the association and object detection. Calibrating and automatically re-calibrating sensors is a reliable sensor fusion system prerequisite. Suitable calibration approaches can be found in [20] and [21].

- **Environmental perception:** Object detection and classification approaches directly rely on the result of the sensor fusion. A varying fusion quality, therefore, implies an inconsistent perception quality. In automated driving, a fluctuating perception quality can, in worst-case scenarios, lead to fatal accidents [3]. Managing the uncertainty of the sensor fusion output is crucial for a reliable perception system.

1.4 Manufacturing Industry

The automotive industry is shifting towards new drivetrain concepts, and Original Equipment Manufacturers (OEM) face changes in their manufacturing processes. Electrical vehicle platforms vary significantly from combustion platforms. This change impacts components used throughout the vehicle and the integration complexity of the overall vehicle. The production technology required to build electric vehicles differs from the one used for conventional drivetrains due to fundamental differences in powertrain concepts. OEMs have a strong focus on quality assurance. While this has worked well for traditional vehicles, electric vehicles, and their components require new processes and technologies. These new technologies introduce the need to leverage expert domain knowledge for monitoring the manufacturing process in an automated way to allow for production scalability. Historically, expert domain knowledge on new technologies specific to electric vehicles has not been required at OEMs and is, therefore, sparse while being crucial for monitoring the quality of the manufacturing process.

Motivation

Electrical motors are one of the critical parts required for the drivetrain of an electric vehicle. The copper windings of a motor are being replaced by a new technology, which

is referred to as hairpins. This new technology improves the efficiency of the motor itself and the production process of the motor [22]. The copper windings of the stator are being replaced by the hairpins, which are thick copper bars. Once positioned, hairpins need to be welded together, and the process is closely monitored to ensure the desired level of quality. Mass-manufacturing these motors requires a highly automated process to identify and mitigate quality problems by reworking defects.

Challenges

Consecutive welding processes have slight variations, which result in minor variations in the welding results. In some cases, the variation causes a defect and needs to be reworked to ensure the desired functionality of the finished product. Laser welding is used to connect the hairpins, which requires more power compared to welding aluminum or steel. An increased laser power results in a higher likelihood of experiencing defects with different characteristics [23]. Identifying and adequately classifying these character defects for rework is a crucial challenge. Expert knowledge is required to correctly identify and classify the defects while automating this process is necessary to address the demand for electrical motors. This has not been properly addressed, and the feasibility of automating the process is crucial for large-scale series production because manual processes to rework components are expensive and time-consuming.

Another challenge is controlling the equipment costs required for process monitoring. 3D scanners and cameras are frequently used for such applications, but there is a significant variance in price for these devices. This directly impacts the unit economics of the electrical motors, which is an important objective for OEM.

1.5 Semiconductor Industry

Rapidly advancing technology in semiconductor Integrated Circuit (IC) manufacturing has impacted social and economic development across the globe. It has allowed ever-greater capacities in processing and storing electronic information at ever-lower costs. [24].

An integral part of every IC is the transistor, an electronic device that amplifies or switches electrical signals or power. As a result, radios, computers, and other electronic devices were made smaller and cheaper. The metal–oxide–semiconductor field-effect transistor (MOSFET) was invented by Mohamed M. Atalla and Dawon Kahng at Bell Labs in 1959. It is the most widely manufactured device of all time. According to [25], an estimated number of 13 sextillions ($13 \cdot 10^{22}$) MOSFETs have been produced until 2018. IC industry growth over the past decades has been driven by doubling component counts every 12 to 24 months, often referred to as Moore's law. The IC industry has shown an average annual growth of 13% over the last two decades [26]. An Intel® 4004 processor from 1971 contained 2300 transistors, whereas a more recent 5^{th} Generation Intel® Core™ processor from 2015 has more than 1.3 billion transistors. The newer chip is 3500 times more performant, costing roughly 60.000 times less [24].

An increased level of complexity requires more advanced processes to ensure a constantly high level of quality throughout the entire manufacturing process. Strict quality assurance is crucial to ensure the desired functionality, especially when using ICs for automation or medical devices. The semiconductor industry faces the challenges of adversaries offering counterfeit electronics at lower prices with strongly varying quality compared to the original products. Moreover, manufacturers may maliciously change chip designs. Because of IC complexity, hardware Trojans pose a significant danger to secure applications since they are hard to detect. Reverse engineering is required for process characterization and counterfeit detection.

Motivation to Automate Reverse Engineering

The semiconductor device fabrication varies depending on the manufacturer and fabrication plants. Manufacturers tend to share little information on the process and the actual dimensions or geometries of the manufacturing process. A single IC is not limited to one specific transistor design, which allows for a large variety of observable technology and process characteristics on a single chip. Known as the technology node, the IC's minimum feature size is considered a characterizing feature of the manufacturing process. The minimum distance of two conductor tracks present on the chip is referred to as half-pitch or half-the-grid spacing. This measure is used by the International Technology Roadmap for Semiconductors (ITRS) as a reference size [27].

The characterization of technology is not exclusively limited to this measure and uses additional parameters to determine the technology:

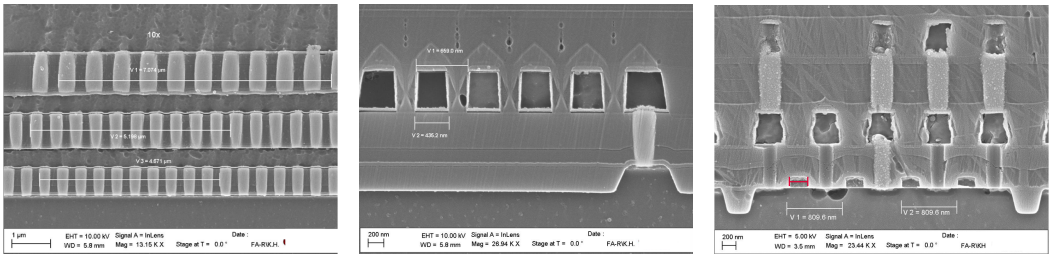
- Contact pitch: Averaged horizontal distance between contacts
- Minimum metal pitch: Minimum horizontal distance between metals
- Minimum gate length: Width of the polysilicon gates
- Gate pitch: Horizontal distance between gates

A visualization of the measurement parameters is shown in figure 1.3.

It is necessary to cut the chip vertically, polish it, and prepare it with chemicals to measure these parameters with a scanning electron microscope (SEM). Images are captured at various zoom levels, which vary strongly depending on the chip packaging and the underlying technology. The identifiable parameters vary by zoom level, as summarized in table 1.1.

Challenges

Several boundary conditions make it difficult to automate reverse engineering. It is necessary to prepare each cross-section individually by hand using a variety of chemicals. As a consequence, different cross-sections are prepared in a variety of ways. The obtained image may only contain partially cut components or the image may be distorted. Recognizing and interpreting the information is considerably more difficult in



(a) Contact (A) and Vertical Interconnection Accesses (VIA) (B) pitch

(b) Minimum metal pitch

(c) Minimal gate length

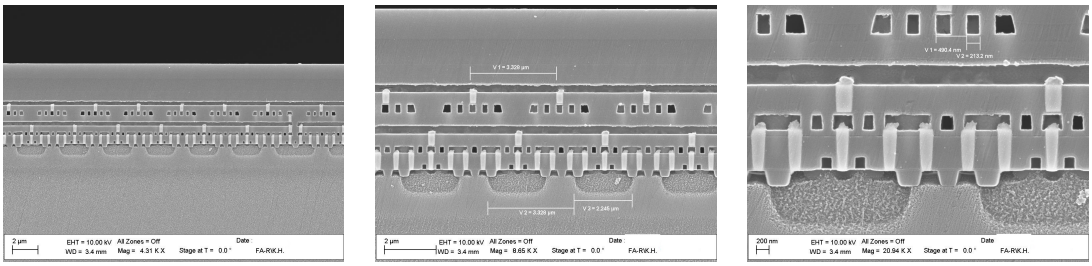
Figure 1.3: Parameters used for technology determination. The horizontal distance between contact pitches is measured and averaged over several elements, as shown in (a). (b) shows the minimum distance between metal layers, and (c) shows the gate length.

Table 1.1: The table shows parameters that can be determined for a given image width and technology size pair. *Metal* is abbreviated as M, *VIAs* and *contacts* as V and *number of* as #. N/A indicates that no useful parameters can be determined.

Technology node (nm)	**Image width** (μm)					
	1	**5**	**10**	**50**	**100**	**>500**
50	M/V Pitches	M/V Pitches	number of M layers	N/A	N/A	Chip thickness
100	M/V Pitches	M/V Pitches	N/A	number of M layers	N/A	Chip thickness
150	N/A	M/V Pitches	N/A	number of M layers	N/A	Chip thickness
200	N/A	M/V Pitches	M/V Pitches	number of M layers	N/A	Chip thickness
250	N/A	M/V Pitches	M/V Pitches	number of M layers	N/A	Chip thickness
300	N/A	M/V Pitches	M/V Pitches	number of M layers	N/A	Chip thickness
350	N/A	M/V Pitches	M/V Pitches	number of M layers & M/V Pitches	number of M layers	Chip thickness

both cases. Additional differences are caused by the use of different microscopes when taking images. For each preparation, multiple images are taken at varying zoom levels, as shown in figure 1.4, resulting in a large variety of images. Components of one class may differ significantly in size, frequency, material, and appearance. As a result, it is challenging to generalize characteristics, even for a human observer. A high-class imbalance results from components that are very small and sparsely appear. A major challenge is demonstrating the feasibility of a methodology for automatic semantic seg-

mentation of microchip cross-section images. An automated reverse engineering process can be enabled with this information.



(a) Magnified 4.310 times (b) Magnified 8.650 times (c) Magnified 20.940 times

Figure 1.4: Cross section of a microchip with different amplification levels.

1.6 Scope and Contribution of this Work

Recent developments in ML and AI have achieved excellent outcomes for consumer products. An ever-increasing availability of computational resources and training data enables this. Providing verifiable NNs at scale is an unresolved challenge. The issue of training data availability for a particular use is currently not addressed adequately in academia or industry. Some applications require large amounts of training data, which is not feasible for many use cases because of the high costs associated with generating the required training samples. These shortcomings of the state-of-the-art are addressed throughout this work. This thesis takes advantage of domain knowledge exploitation by combining knowledge engineering with ML and AI to counteract the drawbacks of individual techniques.

Problem Statement

How can exploiting domain knowledge reduce the architectural complexity of ML-based systems while decreasing the need for training data and computational complexity? Within the scope of this thesis, domain knowledge exploitation for sensor fusion, object detection, classification, and segmentation approaches along with frameworks are established. The underlying hypothesis is that by leveraging domain knowledge, boundary conditions, and constraints can be derived, simplifying the ML problem and the overall system complexity. Based on this hypothesis, the problem statement addressed throughout this thesis can be phrased as follows:

The laws of physics, inevitable conditions, and constraints introduced by the circumstances or processes are the foundation for building a knowledge representation. The engineered knowledge is exploited in a three-step process:

1. Collect available domain knowledge to introduce boundary conditions and constraints
2. Analyze the impact of the knowledge on the use case at hand and identify the most significant lever for reducing the complexity of the overall system
3. Incorporate domain knowledge into the system architecture

Contribution

Existing solutions in the ML and AI domains are not able to simultaneously address the problems stated in the previous section. This work addresses the problem by combining knowledge engineering with ML approaches. The four main contributions of this work are outlined below.

1. The fusion of multi-modal sensor data is crucial for detection and classification tasks. Early-stage data fusion preserves more information for detection, classification, and segmentation tasks. This directly results in improved results for both latency and computational efficiency. The proposed concepts of early-stage sensor fusion leveraging existing domain knowledge benefit the investigated use cases and generalize to different applications across industries.

2. A new approach for a deep multi-modal fusion architecture for heat map-based object detection through segmentation is proposed. It allows for improved segmentation due to early data fusion leveraging human domain knowledge of object appearance. This enables the use of object detection datasets to train semantic segmentation architectures. The early merging of sensor modalities in combination with domain knowledge-driven class appearance models allows the network to associate information more easily, which results in fewer network parameters and simplifies the training process.

3. A new framework for creating a modular multi-stage object classification is developed, allowing for combining human domain knowledge, traditional ML, and deep learning approaches in a single system. Human expertise is used for identifying the desired cascade of classifiers for the desired use case, while modularity allows the retraining of individual classifiers without having to retrain all stages. This enables targeted improvements and retraining in case of required adoption due to slightly changing use cases and creates more transparency compared to a monolithic deep neural network. The different levels of granularity allow for steering the classification depth based on the environmental situation, and, therefore, the computational load can be actively managed.

4. A method for tailoring neural networks to use cases is proposed based on exploiting human domain knowledge of the use case. The limitations of the applications due to boundary conditions do not have to be explicitly learned by a neural network. As a result, the training effort is reduced while the system performance is increased.

A multi-modal sensor fusion is a technical prerequisite for leveraging multi-stage object classification and demonstrating its benefits. The main contributions are demonstrated on use cases from three different domains to showcase the ability to generalize beyond one particular use case. The primary motivation is not to quantitatively outperform specific implementations of NNs. The proposed approaches aim to demonstrate the ability to exploit domain knowledge in combination with traditional ML and deep learning approaches for multiple use cases. This reduces architectural complexity while, at the same time, increasing system transparency and performance.

Structure and Outline

The structure of this work starts with the current state-of-the-art and proposes solutions for sensor fusion and perception or classification, which leverage expert domain knowledge. Chapter 2 provides an overview of the current state-of-the-art technology for sensor fusion, the basics of NN object detection and classification, and segmentation and knowledge graphs. These theoretical concepts are crucial throughout this work and are a prerequisite for the following chapters. Chapter 3 gives an overview of the methodology used for the different use cases described throughout the thesis. In chapter 4, a low-level sensor data fusion for multi-modal systems is proposed, which consumes minimally processed sensor data while leveraging domain knowledge to reduce computational complexity by introducing constraints. The results show low detection latencies and provide a higher level of confidence for object detection. This approach is outlined and leveraged in the following chapters. Chapter 5 evaluates the impact of fusion depth on object detection. A novel approach for training is introduced, which allows utilizing bounding box-based object detection datasets for semantic segmentation. The results evaluate the impact of using human domain knowledge at different fusion levels. In chapter 6, a framework for designing hierarchical classification systems with multiple stages for object classification based on domain knowledge is introduced. An exemplary implementation of the framework on an AD use case is used to evaluate the feasibility of the framework. Chapter 7 develops a digital twin of a rally circuit and simulates multi-modal sensor data under various environmental conditions. The integration of the low-level sensor fusion approach, as introduced in chapter 4, with the multi-stage perception framework from chapter 6 is used to evaluate the overall system performance based on simulated sensor data from the digital twin. The concept of leveraging domain expertise to reduce system complexity is applied to a manufacturing use case in chapter 8. This is an example of a use case using the framework from chapter 6, which demonstrates that the approach is applicable to other domains. In chapter 9, the concepts of knowledge exploitation are applied to a reverse engineering process for semiconductors.

Chapter 10 concludes this work and provides an outlook on the business relevance of the covered topics and future research directions.

2 Literature Review and State-of-the-Art

This chapter provides an overview of basic state-of-the-art concepts that serve as prerequisites for approaches presented throughout this thesis.

2.1 Data Association

Sensor fusion systems are subject to numerous challenges, including data integration. The term "data association" has various meanings in literature, which need clarification in this context. The concept is sometimes called data fusion or fusion of data. Track-to-track fusion refers to the combination of tracks managed by the object tracker. Examples can be found in [12].

Temporal Synchronization

The input data for the association is usually frame-based. Depending on the sensor setup used, the association process has two main issues. The sensor frequency f_{sensor} is defined by the measurement cycles of a sensor per second. As mentioned in chapter 1, different sensors run at different frequencies. Using different sensors results in different frequencies of receiving sensor data. Even sensors of the same technology do not run at the same frequency because they are not necessarily synchronized. Consequently, there is a time difference between individual sensor data frames. In the case of a moving object, it is evident that a temporal difference between measurement frames results in a spatial offset. Therefore, a temporal synchronization method has to be implemented. A good overview and detailed evaluation can be found in [16].

The time delay between the sensor data capturing and the association process is visualized in figure 2.1. Each sensor has a specific latency until data is provided, called sensor latency T_{Sensor} or measurement latency. This comprises the time necessary for data acquisition and pre-processing included in the sensor. The time necessary for transferring data from the sensors to the sensor fusion system is referred to as transfer latency $T_{Transfer}$. Various pre-processing steps in the sensor fusion system introduce delay, which is referred to as system delay T_{System}. Ideally, the timestamping takes place right when the measurements are captured.

Non-Deterministic

The non-deterministic approach is based on sensors working asynchronously and at a non-fixed frequency. Future sensor latencies are unknown, as well as future transfer or

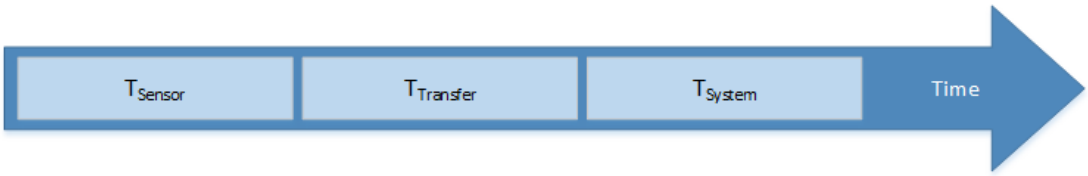


Figure 2.1: Time Delay

system latencies. In this case, the worst-case latency is caused by the association waiting for data from all sensors to be available.

Deterministic

Like the non-deterministic approach, the deterministic approach is based on sensors working asynchronously and at a non-fixed frequency. In contrast to the non-deterministic approach, the deterministic method is able to provide at least the following sensor, transfer, and system latency. The deterministic approach can be used whenever sensors run at a constant frequency. In this case, the association process does not need to wait for the availability of data from all sensors

Synchronized Sensors

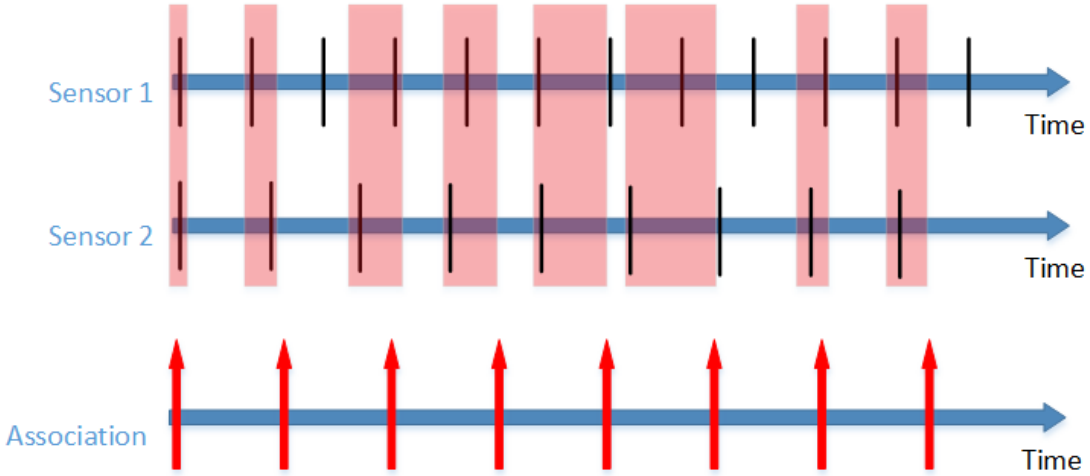
The synchronized sensor approach is deterministic and ideal for association purposes. All sensors can be synchronized to a fixed frequency, which the user can determine. The concept of a synchronized sensor is presented in this work, even though no synchronizable automotive-grade sensor setup is available that covers the desired sensor modalities.

Data-Driven Association

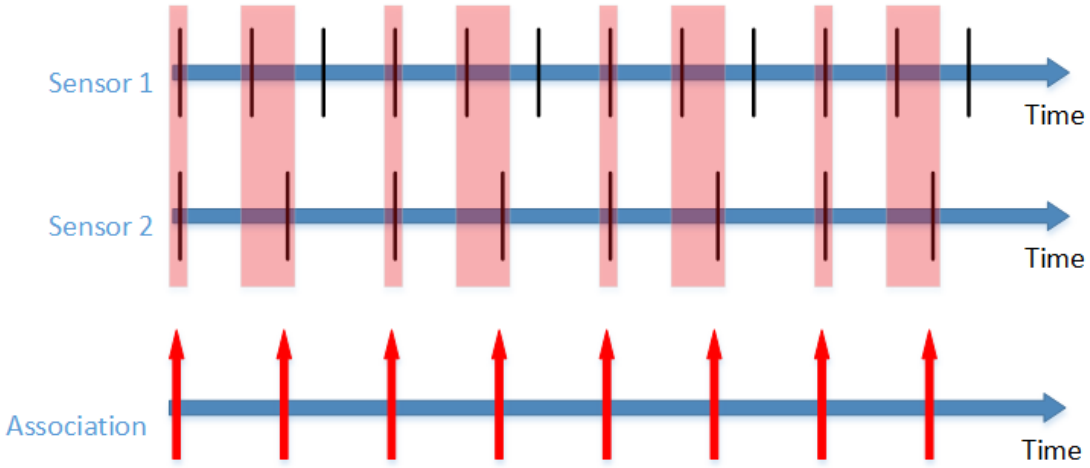
The data-driven association is triggered by the sensor data. One or multiple sensors trigger the association process. Consequently, data is associated whenever new data from a triggering sensor is available. The case of having one sensor triggering the association is depicted in figure 2.2b.

Spatial Synchronization Overview

In addition to the temporal offset between different frames, there is also a spatial offset that has to be compensated to associate data originating from the same physical object. This section briefly describes three state-of-the-art spatial synchronization techniques, all based on tracking.



(a) Time-driven association



(b) Data-driven association

Figure 2.2: Time-Driven and Data-Driven Association: The measurement frames from the sensors are shown in black. The red arrows indicate the time of association. The red area marks the measurements, which are used for association.

Camera Projection

One problem that has to be solved is associating data from a camera sensor producing 2D image data with 3D data from other sensor modalities. Therefore, the camera pinhole model is used, which is explained in detail in [28]. In the scope of this work, objects detected in the 2D image space are projected as a set of rays into the orthogonal projection of the 3D environmental model, as shown in Figure 4.12. A method of creating 3D information from a 2D source, such as depth maps or a structure from motion, is not part of this present work.

State-of-the-Art Spatial Association

In [29], a data association method for fusing RADAR, LiDAR, and the camera is proposed. Advanced tracking models are necessary for this approach, including motion and observation models for each sensor modality. To be able to use the correct object models, an object classification is required. This means that the classification is performed before association and tracking.

In [30], another approach for spatial alignment is proposed. This approach also depends on tracking. In this particular implementation, the history of each track and, therefore, the individual object hypothesis history is maintained. Based on the previous states, the future state of an object is predicted. The object hypothesis is matched to the closest track using the nearest neighbor method.

A more straightforward approach towards spatial association is matching reference points as described in [31] and [32]. Such a reference point can be the geometric center or the center of gravity of a bounding box. Again, a tracker is used to track the reference points and to predict the future position of such points. The goal is to find the most suitable prediction for a current reference point. A more detailed description of such methods can be found in [31] and [32].

Sensor Fusion Approaches

An increasing number of sensors in modern vehicles and the desire to achieve fully automated driving create the need to combine information from various sources into one consistent environmental model.

Different sensor modalities produce distinct data outputs that have to be fused. This can be accomplished by using sensor fusion approaches which combine the available information to achieve a higher level of detection confidence. There are numerous approaches depending on which sensors are used and the kind of sensor setup. Some sensors include processing logic and are called smart sensors for that reason. Those sensors pre-process the raw data and output either a feature, a list of targets, or a list of objects. For example, some RADAR sensors output a list of confirmed objects. This means that a target must be tracked for multiple frames to confirm that the measurement correlates to a real object and not to a ghost object, such as a street reflection [33]. This procedure introduces additional object detection latency. Depending on which level the informa-

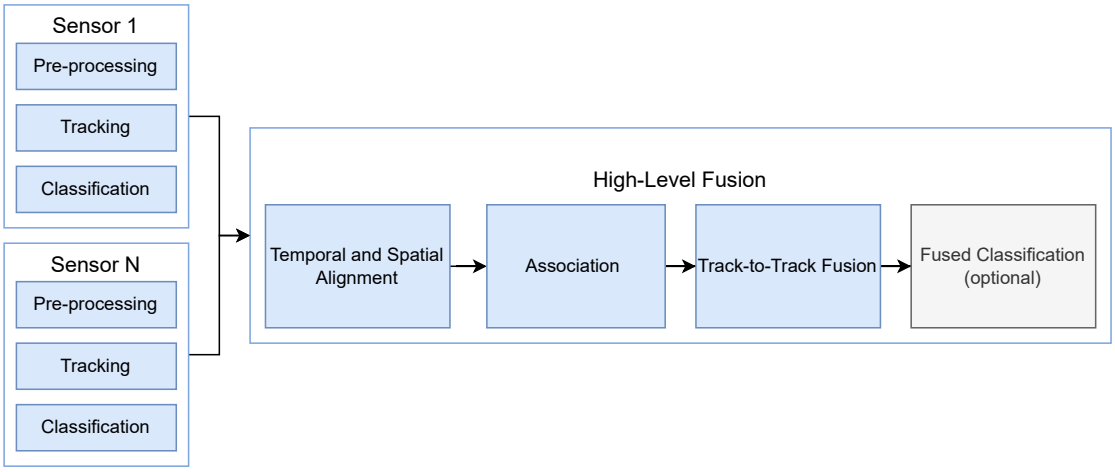


Figure 2.3: High Level Sensor Fusion

tion from different sensors is fused, one has to differentiate between the sensor fusion concepts described in this section.

High Level Sensor Fusion

Figure 2.3 shows a high-level sensor fusion system. Each "smart" sensor individually captures data, pre-processes it, and tracks potential objects. In this context, pre-processing is, for example, image signal processing for automatic white balancing in a camera. Tracking objects requires multiple consecutive frames. If noise or clutter results in an object track being lost, the tracker may fail to maintain the track. Under these circumstances, reinitialization of the tracking is necessary On the other hand, successful object tracking allows outputting a list of confirmed objects with associated tracks. This list is sent to a high-level fusion system, where the sensor data is spatially and temporally aligned and associated. After the association step, a track-to-track fusion is performed, which can be followed by a classifier.

Advantages

- Sensor fusion via independent detections from each sensor
- Low requirements for communication bandwidth
- Independence from specific sensor vendors

One benefit of this method is that sensor-independent processing is guaranteed until the data from the individual sensors are associated. The final system decision is based on independent object detections and the corresponding tracks.

Another benefit is a low bandwidth for communication between the sensors and the sensor fusion system. Only object data and associated tracks have to be transferred. The bandwidth is lower compared to feature or low-level sensor fusion approaches.

The independent choice of sensors is an advantage because the information is combined at the object level. Signal processing of sensors can be very specific and is performed in the smart sensor. The association and fusion happen on a high abstraction level. Therefore, sensor interfaces can easily be adapted to match the desired object description. Thus, objects can be fused. This introduces a low sensor vendor dependence.

Disadvantages

- Loss of information
- Increased latency

Smart sensors output detected objects. The captured data has to be good enough that an object can be successfully detected in just one sensor modality. In some cases, individual sensors do not have sufficient data to detect an object, even though an object is present. Consequently, in such a case, a high-level sensor fusion system does not have data for the fusion; therefore, information is lost.

Latencies in processing are introduced in several ways. First of all, the sensor-specific pre-processing consumes time. Additionally, the tracker introduces more latency, as a couple of frames are necessary for successful tracking. The latency becomes very significant when a track has to be re-initialized.

Feature and Object Level Sensor Fusion

Figure 2.4 depicts a feature and object-level fusion system. The captured data is pre-processed, and features are extracted. Pre-processing can, for example, be noise or clutter removal from street reflections in a RADAR. After extracting the features, they are associated, and objects are generated. The generated features and objects are sent to the fusion system, where data from different sensors is temporally and spatially aligned. The provided features and objects are fused after the association step. An optional object classifier can follow. In [34], an implementation of a possible feature-level fusion can be found.

Advantages

- Utilization of object and feature data for fusion
- Enhancement of system robustness

Objects and their features are provided to the fusion system, which provides valuable information for object detection and classification. Features are more discriminating and provide more information than object data only.

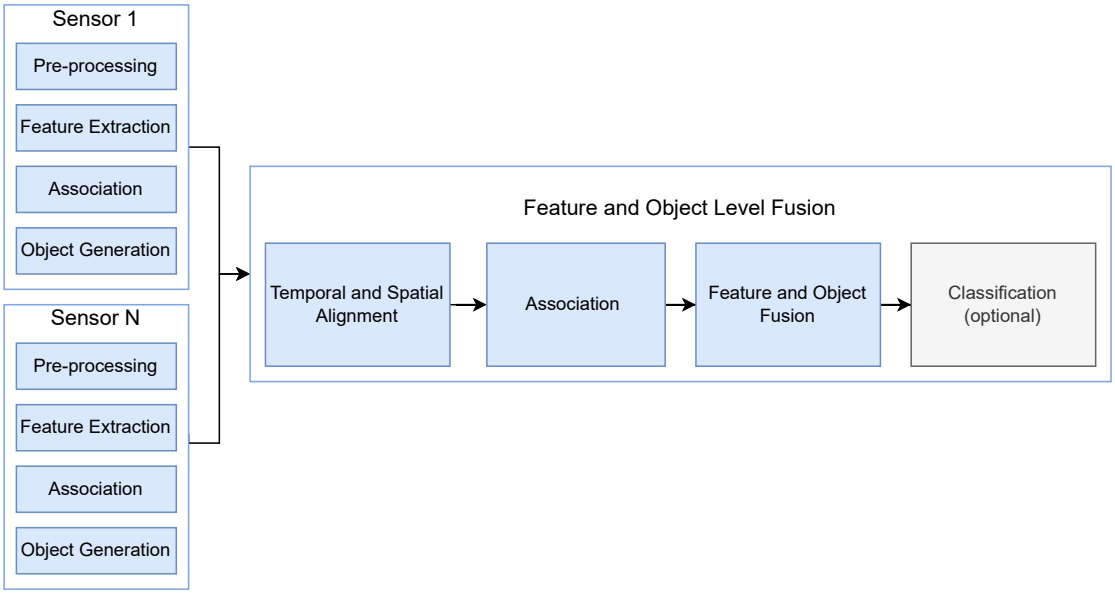


Figure 2.4: Feature and Object Level Sensor Fusion

Data captured by a sensor might not be sufficient to generate an object, but the detected features are available for fusion. The final object detection is then based on untracked features and object data from the sensors. Therefore, the fusion is more reliable and robust than a high-level sensor fusion.

Compared to a low-level sensor fusion system, an advantage of this approach is a lower communication bandwidth.

Disadvantages

- Necessity of object models
- Dependence on specific sensor vendors
- Risk of information loss

A disadvantage of a feature-level fusion is the dependency of the fusion system on feature extraction. The result of the fusion strongly depends on the available features. Because of this fact, sensor vendor independence is harder to achieve.

Additionally, sensors have to use feature-based models to generate objects. In 2D and 3D space, different models have to be used. This results in uncertainty when combining objects generated in different spaces and again limits the choice of sensors, as the models have to be compatible.

In comparison to a low-level fusion method, one further downside is the loss of information resulting from feature extraction. Regardless of whether the feature definition is

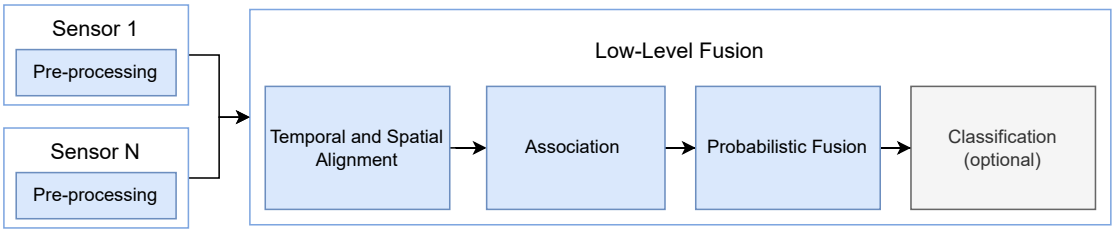


Figure 2.5: Low Level Sensor Fusion

simplified to a cluster of raw sensor measurements, individual sensors will disregard clusters that lack sufficient density. This is a contrast to the technique detailed in chapter 4. Unfortunately, this disregarded information is not forwarded to the feature fusion, where it could provide valuable insights when paired with observations from other sensors.

Low Level Sensor Fusion

Figure 2.5 shows a low-level sensor fusion system. The sensors are directly coupled to the fusion system, which requires a high-speed link as the amount of transferred data is very high. The captured and unprocessed sensor data is referred to as raw data. Before executing the association process, all sensor readings need alignment both spatially and temporally. Once this alignment is complete, the readings undergo probabilistic sensor fusion, which is then followed by a classification step.

Advantages

- No loss of information
- Reduced latency
- Increased robustness
- Sensor vendor independence
- Scalable sensor data processing chain

From an information theoretical perspective, a low-level sensor fusion is optimal because no information is lost [35]. All captured information is preserved and provided to the fusion system. The detection decision is based on the fused data from all sensors. As a consequence, all available information is preserved. This also maximizes detection confidence and increases the reliability and robustness of the system.

The latency between data capturing and the sensor fusion system is reduced because the direct coupling of sensors does not introduce sensor-specific processing delays. A centralized processing platform has to be designed so that the latency toward the object detection phase is further decreased.

The raw data does not depend on the sensor vendor, and no sensor-specific signal processing is necessary. The quality of the data varies for different sensors. For example, a LiDAR returns a point cloud, but the number of points and, therefore, the density depends on the used sensor and its specifications.

Disadvantages

- High data rates
- No sensor-independent detection results

A disadvantage of low-level sensor fusion is the large communication bandwidth required to transfer raw-level data from the sensors to the fusion system.

The detection is based on the fused information from all sensors. Consequently, there is no independent decision from individual sensors available.

Multi Level Sensor Fusion

Depending on the available sensors, combining all information on the same level is not always possible. The available raw data is fused; thus, all relevant information is provided to the sensor fusion system. Other sensor modalities, providing data at a higher level, can be associated with the features extracted from raw-level data. A multi-level fusion system fuses data on different levels, always using the lowest available level for individual sensors. All information provided by sensors is used, and no information is discarded. For example, RADAR and LiDAR data are associated at a raw level, and features are extracted after the association process. The extracted features are then fused with features provided by a camera sensor. This results in an optimal use of the available information, which can be used for object detection.

2.2 Fusion Networks

Statistical inference is defined as the process used to create hypotheses about underlying distributions and their parameters by analyzing data. There are different approaches towards inference and even various schools, such as the ones mentioned in [36]. In order to represent probabilistic knowledge, it is a common approach to use numerical representations. A graphical representation is more intuitive and human-readable. A graph structure in which the nodes represent propositional variables and edges represent dependencies is a common model of a joint probability distribution. Depending on the choice of graph, dependencies and independencies can be visualized using undirected or directed graphs.

Bayesian and Markov Networks

In this thesis, Bayesian and Markov networks are analyzed, as those are well-understood and popular methods for uncertainty management. Factor graphs can be seen as a further development and combination of those approaches [37]. A Bayesian network is a

directed graph and allows for capturing more probabilistic dependencies in comparison to a Markov network, which is an undirected graph. The fact that the Bayesian approach allows capturing conditional independencies is advantageous. A more detailed comparison of Bayesian and Markov networks can be found in [36] and [38]. Based on these analyses and the fact that a well-understood method is desired for the present work, Bayesian networks are chosen as a fusion network. Exact inference is chosen as a tool for evaluation purposes, as it is a deterministic and simple inference method.

Assuming that the decision problem is posed in probabilistic terms, the Bayesian decision theory is one of the most fundamental statistical approaches. Therefore, the concept of Bayesian inference is briefly explained in this section.

Basic Concept

Using Bayesian methods as the underlying knowledge representation, the following basic rules apply, which can be found, for example, in [36] or [39].

A set S contains all possible events A. Therefore, the following basic rules of probability theory apply:

An event has to have a probability between 0 and 1:

$$0 \leq P(A) \leq 1 \tag{2.1}$$

The probability of any event happening is equal to 1:

$$P(S) = 1 \tag{2.2}$$

The conditional probabilities are defined as the following:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2.3}$$

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \tag{2.4}$$

In general, the Bayes theorem describes the conditional probability of a parameter, given an observation. The to-be-determined conditional probability, in this case, is the posterior probability $P(A|B)$ of A, given the observation B:

$$P(A|B) = P(B|A)\,\frac{P(A)}{P(B)} \tag{2.5}$$

Now, based on these basic rules, a set of N events A_n is defined:

$$A_1, A_2, ..., A_{N-1}, A_N \tag{2.6}$$

The joint probability of the set of variables is defined as the probability of a specific value assigned to all the variables in the set. The joint probability is therefore defined as:

$$P(A_N, ..., A_1) \tag{2.7}$$

Applying the Bayes rule on the joint probability yields the following formula:

$$P\left(A_N, ..., A_1\right) = P\left(A_N | A_{N-1}, ..., A_1\right) P\left(A_{N-1}, ..., A_1\right) \tag{2.8}$$

Splitting the last term of this equation and repeating splitting of the last term of the respective resulting equation results in a chaining process called the chain rule, which is defined as:

$$P\left(\cap_{n=1}^{N} A_n\right) = \prod_{k=1}^{N} P\left(A_k | \cap_{j=1}^{k-1} A_j\right) \tag{2.9}$$

In this example, the events A, B, and K are given. A and B are conditional independent given K if the following holds:

$$P(A|B,K) = P(A|K) \tag{2.10}$$

A simple Bayesian network is visualized in 2.6, showing the relationship of the events A, B, and C. As this is a directed graph, the conditional independence of events B and C is obvious. Let node A be the existence probability of an object. Node B determines whether the object is detected by a sensor, for example, LiDAR. Finally, node C indicates whether an object is detected by another sensor, for example, the camera sensor. It is obvious from the graph structure that nodes B and C certainly influence the existence probability, but the degree to which they influence it is not contained in the graph structure. The contribution amount is stored in a conditional probability table, like table 2.1.

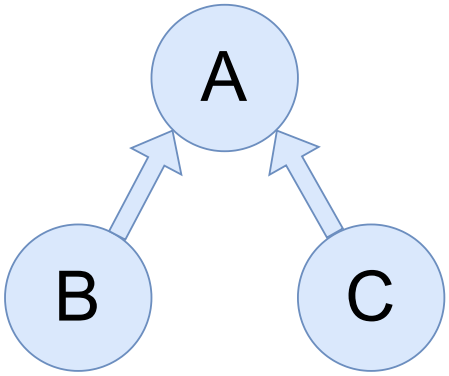


Figure 2.6: Simple Bayesian Network: B and C are the leaf nodes. They are conditionally independent given A, and A is the root node.

Choice of Fusion Network

This work focuses on a low-level sensor fusion approach, which can be used on embedded hardware in safety-critical environments. For evaluation purposes, an approach is desired which can be efficiently implemented without extensive development effort. The fusion

Table 2.1: Conditional Probability Table for Bayesian Network

	B true C true	B true C false	B false C true	B false C false
A true	0.99	0.9	0.7	0.05
A false	0.01	0.1	0.3	0.95

network is supposed to run on real sensor data provided to the system at different frequencies. Based on the comparison in 2.2 and the following requirements, the Bayesian network is a suitable choice for a fusion network:

- Provision of intuitive and human-readable knowledge representation
- Availability of many well-known and efficient exact or approximate inference implementations
- Capability to handle different node update and node query rates efficiently
- Easy extension into a dynamic Bayesian network for stateful inference
- Expression of conditional independencies
- Applicability on embedded hardware
- Characteristic of being a deterministic network

2.3 Basics of Neural Networks

Basic Architectures

Multi-layer Perceptron

In the 1980s, MLPs were a popular ML solution for image recognition. The input is a vector. For an image to be used as input, its pixels must be flattened. Due to this limitation, MLPs are unable to identify objects in a picture, regardless of their position within the image. The same object located at a different location in the picture would have a completely different input vector [40].

Feed-Forward Neural Networks

A neuron aggregates weighted input activity and passes it through to the output. Each element of the input x and the bias b is multiplied with a weighting factor w and summed up to z. Depending on the sum z, an activation function determines if a neuron fires and how strongly it fires. A perceptron uses a step function as an activation function, which results in the perceptron firing if the sum z exceeds a certain threshold. This means that a small change in input can directly result in a significant difference in output behavior.

In contrast, a neuron uses a linear or non-linear function as an activation function. This achieves the desired property of having a network where a slight change in the weights causes a small change in the output. Figure 6.1 provides an illustration of a feed-forward NN.

A neural network is a collection of neurons that are arranged in layers. The layers between the input and output layers are referred to as hidden layers. A neural network learns the relationship between inputs and outputs by creating a numerical model based on example input and output pairs as part of a supervised learning process. This approach enables pattern recognition and data classification tasks based on annotated data. In traditional non-ML-based approaches, expert domain knowledge would be required to derive a traditional heuristic. When training a neural network, input and output values are labeled and human-comprehensible, thus, reflecting the domain knowledge. The hidden layers are trained by adjusting the weights, and biases of the connection between neurons and the hidden layers are not observable during inference. In a feed-forward network, the sum z at a neuron depends exclusively on the activation set of the neurons of the previous layer and a bias. Such networks are referred to as feed-forward multi-layer neural networks.

Convolutional Neural Networks

CNN are a widespread architecture, and they are based on neurons with learnable weights and biases, comparable to feed-forward networks. The resulting network approximates a transfer function and turns input tensors, which are multi-dimension matrices, into class scores. A CNN uses a combination of convolutional, pooling, and fully connected layers to transform an image into class scores while typically using a loss function on the final, fully-connected layer to determine the class scores. However, CNNs exploit the fact that the input is images which allows the creation of a more efficient feed-forward architecture, reducing the number of parameters used in the network. This is based on the assumption that the provided input is images. The underlying hypothesis is that an image can be separated into overlapping local regions without losing the contextual information required for determining the desired output classes.

Convolution layers comprise a set of learnable filters with a defined width and height. These layers transform local volumetric regions in the input data to output by forming the dot product of the input of a region at any position and the convolutional filter, as explained in 2.3. The result of sliding the filter over the input is an activation map. During the training process, the filter weights are adjusted so that the activation maps react to a specific input feature, which could, for example, be an edge in the image. The deeper the layer in a CNN is, the more complex the relationship between activations and input patterns gets. Each filter in a convolutional layer creates an activation map, and all activation maps of layers are stacked to generate the output of the layer.

Since neurons are only connected to local regions, the split into regions results directly in local connectivity. This connectivity and its extent are generally referred to as the receptive field of a CNN. Regarding the depth of the input, the connectivity in the 2D space is local and full. Compared to fully connected networks, where a neuron is

connected to all neurons of the previous layer, this is a significant difference of CNNs, which is the main reason for a network parameter reduction.

A more detailed introduction to CNNs is given in [41].

Basic Building Blocks of Neural Networks

Pooling

Downsampling is the process of creating a lower-resolved version of an input signal. Structures that are important to the task at hand are preserved, while irrelevant details are eliminated. As a result of downsampling, the number of parameters is reduced, resulting in increased computational efficiency. The translational equivariance is obtained by using downsampling. Therefore, a small translation in the input does not affect the output.

With a convolutional layer, downsampling can be accomplished by varying the stride. The output feature map will have half the size of the input tensor when the step size is two.

In most cases, downsampling is performed by pooling layers. Pooling layers are commonly added after nonlinearities like the Rectified Linear Unit (ReLU) layer. This method is similar to applying a convolutional filter to the input feature maps. Each feature map is pooled separately by the pooling layer to create a new set of pooled feature maps. The pooling operation is specified, and two common functions are used: Average pooling and maximum pooling. The average or maximum value is used for each patch on the feature map. A window function is applied to the input, which provides the viewport content to the pooling function [42]. The pooling operator or filter must be smaller than the feature map. A frequently used pooling operator size is 2x2 pixels while using a two-pixel stride. This reduces the input feature map by a factor of two. Due to the reduction in both directions, horizontally and vertically, the resulting reduction in values is fourfold.

Three pooling operations are commonly used in CNNs as shown in figure 2.7 and explained in more detail in [42]:

- **Average pooling:** Calculates the average of the pixels inside the receptive field on the feature map.
- **Max Pooling:** Takes the maximum value of the pixels inside the receptive field of the feature map.
- **Global Pooling:** Reduces the feature map to one value using average or max pooling.

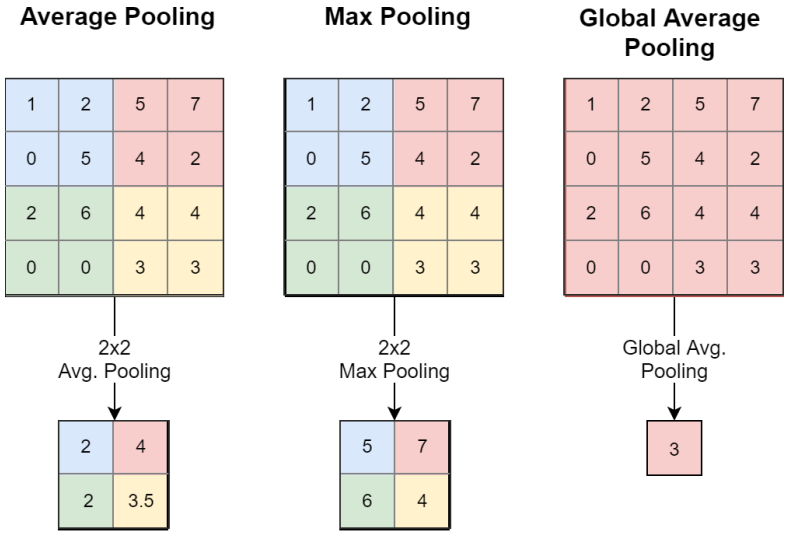


Figure 2.7: Three commonly used pooling operations in CNNs: Average Pooling, Max Pooling, Global Average Pooling.

Most algorithms for semantic segmentation use an encoder/decoder structure, where the low-level feature maps generated by the encoder are upsampled into a higher-resolution feature map.

Upsampling is easiest if the input is scaled to the desired size, after which an interpolation method is used to calculate pixel values at each point. Commonly used here are a nearest neighbor, bilinear, and bicubic interpolation methods. Deconvolution and unpooling are other, more complex methods.

Unpooling

Through unpooling, the resolution is upscaled by distributing a single value across a larger range. In the context of CNNs, reversed max-pooling, sometimes referred to as unpooling, is frequently used. Despite the non-invertible nature of the maximum pooling operation, an approximate inverse can be computed by keeping track of the maximum locations within each pooling region via a set of switch variables. As seen in the DeconvNet study, these switches assist the unpooling operation by assigning the reconstructions from the preceding layer to their proper positions, thereby maintaining the stimulus structure. This is illustrated in Figure 2.8. This method stands out from other techniques by transmitting a greater amount of information to feature maps with higher resolution [43, 44].

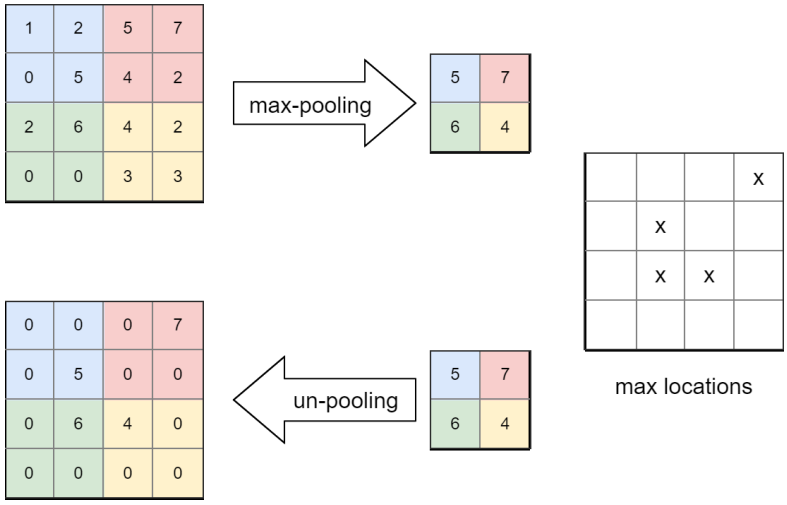


Figure 2.8: Reversed max-pooling. The locations of the maxima are recorded in order to have an approximation of the inverse of the max-pooling operation.

Convolutions

The convolutional layer enables efficient image processing by transforming images into an alternate representation. To achieve this, features are extracted, limiting the loss of relevant information. A predetermined number of filters are applied to the input matrix, each of which has a fixed size, for example, 2x2 or 3x3. The input matrix is then convoluted with these filters.

Convolution occurs by sliding a window over the input pixels with a constant step size over the input tensor. The product is computed between each element of the filter and the input element it overlaps with at each location. Those results are added together to get the output at the current location, as shown in figure 2.9 [42]. he behavior of the filter upon reaching the tensor boundary is dictated by the padding scheme. It is important to note that the size of this resulting tensor depends on the size of the filter, the padding, and the step size [42].

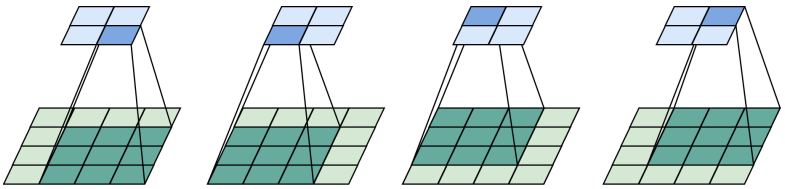


Figure 2.9: Convolution of a 3x3 filter (dark green) over a 4x4 input (light green) without padding and unit stride. The resulting feature map is a 2x2 matrix (blue). Each step of the convolution is highlighted in dark blue in the output matrix.

Transposed Convolutions

A transposed convolution, also known as a deconvolution in the context of CNNs, does not refer to a mathematical deconvolution in the conventional sense. To better understand the transposed convolution, it is necessary to recognize that the operation can be represented by a sparse matrix C. You can achieve this by unrolling the input and output into vectors from left to right and top to bottom. The non-zero elements $w_{[i,j]}$ of the sparse matrix represent the weights of the kernel, where i and j specify the row and column of the kernel. Figure 2.10 shows a 3 x 3 kernel convoluted against a 4 x 4 input using unit stride, generating a 16-dimensional vector at each step. Stacking these vectors results in a 4x16 matrix C in which each row represents a convolution operation. Using it requires flattening the 4x4 input matrix into a 16x1 vector. Matrix multiplication of the convolution matrix C and the flattened input results in a 4x1 output matrix, which is reshaped into the desired 2x2 matrix. [42].

$$C = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix}$$

Figure 2.10: Convolution matrix C of convolution in figure 2.9. The convolution can be represented as a sparse matrix of size $4x16$, where each row represents a step of the convolution operation. The non-zero elements $w_{i,j}$ represent the weights $w_{i,j}$ of the filter.

Obtaining a backward pass using this representation by transposing C is straightforward. Consequently, the error is calculated by multiplying the loss by C^T. A four-dimensional input vector is used in this operation, and a 16-dimensional output vector is produced.

A transposed convolution maintains the connectivity pattern of the original convolution while seeking a one-to-many relationship. Transposed convolutions accomplish this by swapping the forward and backward passes of the convolution.

2.4 Object Detection and Classification

Object Detection

In recent years many different architectures for object detection and object recognition have evolved, and significant improvements concerning performance have been achieved. The most common architectures for object detection are single-stage and two-stage detection networks. Single-stage networks, like SSD [45], or YOLO [46], combine object detection and object classification in a single stage, whereas two-stage detection networks are more complex and separate the object detection from the object classification task. Two-stage architectures are known from R-CNN [47], and architectures evolving from it like Fast R-CNN [48], Faster R-CNN [49], or Mask R-CNN [50].

Object Classification

Image classification challenges, like ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [51], strongly impacted the development of classifiers. Support vector machines and nearest neighbor classifications were the methods of choice before neural networks experienced their revival. Since then, many network architectures have been developed, like AlexNet [52], VGG [53], Inception [54] and Residual Neural Network (ResNet) [55]. These networks employ a range of methods to tackle common challenges faced by data-driven learning approaches. Some of these are over-fitting a classifier to the used training data and vanishing gradients. The authors of AlexNet applied dropout layers as a method to prevent over-fitting, whereas the authors of ResNet used residual blocks to address the vanishing gradient problem. All networks have in common that they use convolutional kernels to extract features based on which they classify. The size of the used kernels also varies for these networks. AlexNet uses different-sized kernels, whereas VGG only uses $3x3$ kernels.

2.5 Segmentation

This section offers a summary of segmentation fundamentals. It delves into individual architectures and encoders, highlighting their unique characteristics. More details on additional architectures and encoders can be found in the appendix A.

Encoder-Decoder Based Segmentation

The concept of convolutions was proposed for solving visual pattern recognition tasks as early as the 1980s [56]. However, the current application of convolutions for image recognition is the result of much more recent achievements. In 2012, AlexNet, a deep CNN, surpassed the state-of-the-art in ImageNet image classification performance by over 10% and revolutionized the field of computer vision [57]. The CNN architecture is today one of the most commonly used architectures within the deep learning community, particularly for computer vision tasks.

With the assistance of a CNN, an image is converted into a feature vector. The input to a CNN is represented as a tensor. As a consequence, the network is able to analyze images that are represented as tensors with three channels: width, height, and color channels. A CNN uses a series of filters and downsampling strategies to encode images into feature vectors. Fully connected layers process the generated feature vector to perform classification.

Semantic segmentation involves upsampling the feature vector either with deconvolutions, by unpooling layers, or directly by interpolating to recover its original shape. An output layer with a sigmoid function is then used to assign each pixel a class.

The downsampling and upsampling structure forms an encoder-decoder architecture that enables semantic segmentation. An example is shown in figure 2.11. Throughout this section, the building blocks of this network architecture are described.

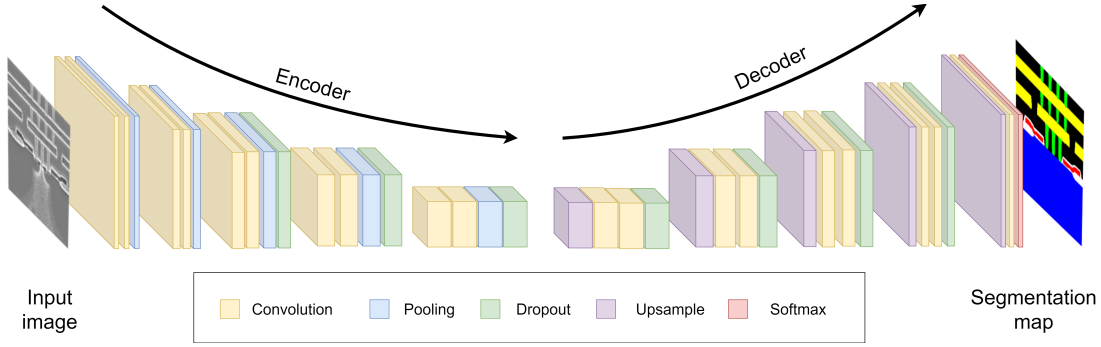


Figure 2.11: General structure of an encoder-decoder architecture used for semantic segmentation. It uses a series of convolutional layers (yellow) to extract features. The feature map is downsampled in the encoder with pooling layers (blue). The decoder upsamples it (violet) to recover its original shape. Finally, a softmax layer (red) generates the segmentation mask. Dropout layers are used to avoid overfitting (green).

Encoders

VGG

At the ILSVRC-2014, Oxford's Visual Geometry Group (VGG) presented its deep convolutional network for object recognition. The challenge relies on a database of more than 14 million images organized into 1000 classes. Convolutional filters with a receptive field of 3x3 are applied to each image through a stack of convolutional layers. The convolutions are stride one with the same padding to preserve spatial resolution. In order to reduce the feature maps, certain convolutional layers are followed by max-pooling layers, as shown in figure 2.12. Maximum pooling is executed by using 2x2 pixel windows with a stride of two. Every time the size of a feature map is halved, the depth of the map is doubled. On the basis of the feature maps, three fully connected layers are used to generate the classification output.

Rather than using a few layers with big receptive fields, a stack of convolutional layers with small receptive fields is used in the first layer. This results in fewer parameters and greater nonlinearity. As a result, the transfer function becomes more discriminating and easier to learn. Convolutional layers can be stacked at different depths, resulting in deeper or shallower networks. Depending on their stack depth, these architectures are in the literature referred to as VGG-11, VGG-13, VGG-16, and VGG-19 [53].

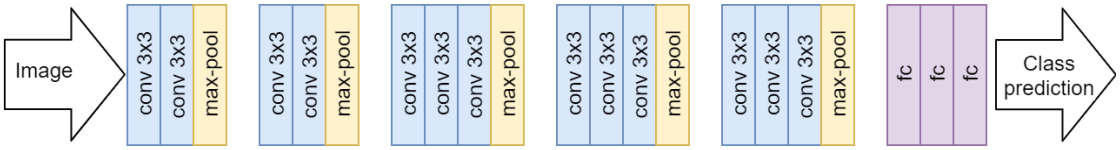


Figure 2.12: Architecture of VGG-16. Several identical 3x3 convolutional layers are stacked. The size of the feature maps is gradually reduced with 2x2 max-pooling layers while the depth of the feature maps increases. Three fully connected layers followed by a softmax classifier generate the prediction.

Inception

Inception Network tackles the problem of significantly varying object sizes. Although larger kernels are better at handling global information, smaller kernels can handle local information more effectively, which needs to be considered when selecting the kernel size. Since stacking layers is computationally expensive and prone to overfitting, it is not an option. Therefore, Inception Network implements multiple sizes of filters operating at the same level, and instead of stacking more layers, it widens them. The inception block includes three different filters corresponding to 1x1, 3x3, 5x5, and a max-pooling layer, and the resulting feature maps are concatenated. Convolutions of size 1x1 are introduced before the 3x3 and 5x5 and after the max-pooling layers to reduce computational costs.

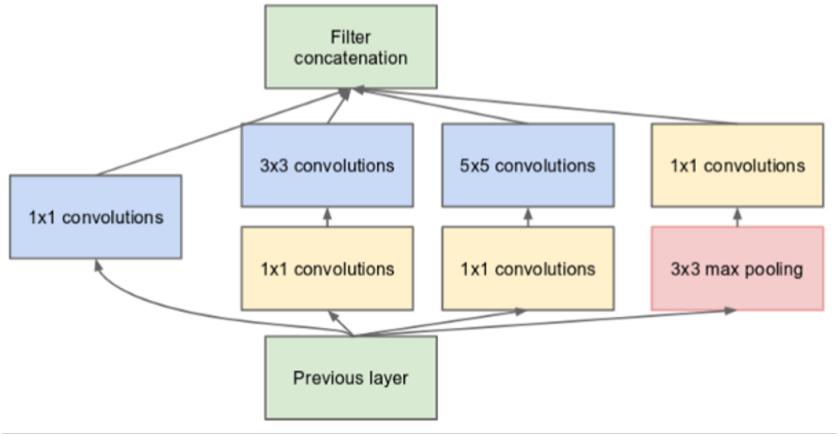


Figure 2.13: Inception module

Residual Neural Network

A ResNet is a neural network composed of 152 layers and has eight times the depth of VGG-19 while being less complex. ResNet introduced a type of building block, the residual block, based on identity shortcut connections that bypass one or more layers and perform identity mapping. Figure 2.14 shows a residual block. Its output is added to the output of the stacked layers. It is important to note that shortcut connections do not add any extra parameters or add to computational complexity. There are two major design principles that ResNet adheres to:

1. If the size of the feature map does not increase, neither does the number of filters in the layer.

2. If the size of the feature map is halved, the number of filters in the layer is doubled.

The network preserves time complexity, and in contrast to other architectures, down-sampling is done using convolutions with a stride of two [55]. The vanishing gradient problem can be overcome through skip connections, allowing for the training of very deep networks.

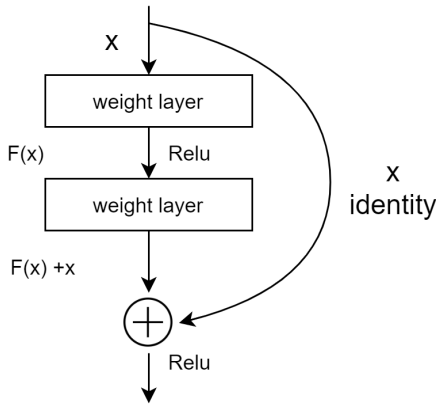


Figure 2.14: Residual block: Skip connections are used to skip layers. In this case, a double-layer skip is shown. It uses non-linearities (ReLU) in between layers.

DenseNet

In DenseNet, skip connections are exploited more extensively since each layer is directly connected to all other layers. Every layer receives as input the output of all previous layers, while the output of each layer is passed to all subsequent layers, as depicted in figure 2.15. Concatenation of feature maps avoids the vanishing gradient problem while allowing for highly efficient use of parameters due to the reuse of features.

A layer l has $k * (l-1) + k_0$ input feature maps, where k_0 is the number of channels of an input image. The growth rate hyperparameter k allows tuning the network width. 1x1 convolutions are leveraged as bottleneck layers to reduce the number of features provided to 3x3 convolutional layers [58].

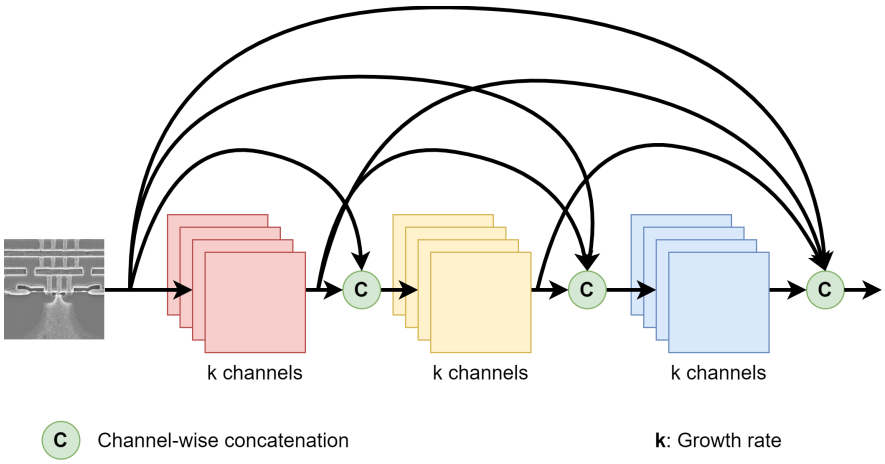


Figure 2.15: Denseblock in DenseNet with growth rate k. Each channel receives feature maps from all preceding layers.

Encoder-Decoder Architectures

A large variety of architectures have been introduced to achieve pixel-level classification, and a comprehensive overview of more than 100 significant state-of-the-art deep learning-based segmentation models can be found in [59].

Fully Convolutional Network

Semantic segmentation is often performed by using Fully Convolutional Network (FCN)s. By replacing all the fully connected layers with convolutional layers, existing architectures are modified to have a non-fixed-size input. The FCN utilizes deconvolutions to upsample several feature maps of smaller sizes. Skipping connections allow high-level features to be combined with low-level ones.

U-Net

U-Net was introduced for biomedical image segmentation and surpassed state-of-the-art approaches as demonstrated in [60]. U-Net is built upon the FCN architecture while having major differences compared to FCNs:

- U-Net has a symmetric setup: a contracting encoder path captures context, followed by an expanding decoder path, thereby enabling better localization.

- Skip connections between the downsampling and upsampling paths are concatenated to preserve features from both paths.

It consists of several stages beginning with two 3x3 convolutions, followed by a ReLU and a max-pooling operation of 2x2 with stride two. Feature channels are doubled with each downsampling step in the encoder. At the same time, the decoder uses transposed convolutions to upsample the feature maps, halving the number of feature channels with each upsampling step. The respectively resulting upsampled feature map is concatenated with the corresponding feature map of the encoder. After the decoder, two 3x3 convolutions and a ReLU follow. The architecture is shown in figure 2.16.

Using an encoder-decoder structure with skip connections allows for the recognition of sharper object boundaries since it gradually recovers spatial resolution while refining the feature map with skip connections.

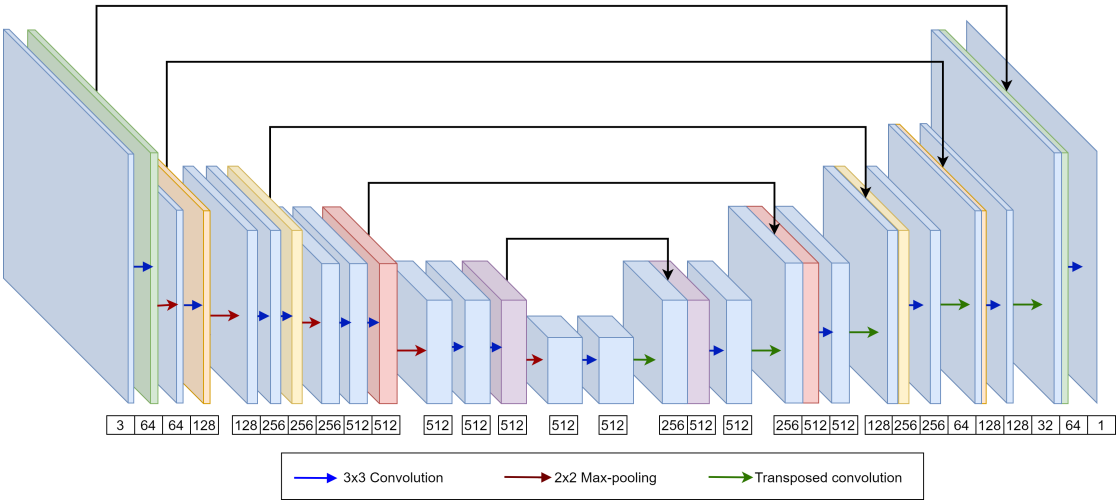


Figure 2.16: U-Net architecture. Each multi-channel feature map is represented by a blue box. The number of channels is denoted at the bottom of the box. It features skip connections that concatenate the output of one stage in the encoder with the feature map on the corresponding level in the decoder (both share the same color in the graphic). The arrows denote convolutions (blue), max-pooling operations (red), and transposed convolutions (green).

Pyramid Scene Parsing Network

Pyramid Scene Parsing Network (PSPNet) contains a Pyramid Pooling Module (PPM) that incorporates global information enabling the segmentation of images. The feature map of the final stage of an encoder is passed through the PPM. PSPNet can capture rich context information by extracting features at varying scales. The fusion of features is based on four pyramid scales as shown in figure 2.17. The levels are:

- Red: Global pooling produces a single bin of output at the coarsest level
- Yellow, Blue, and Green: Splits the output into different sub-regions of 2x2, 3x3, and 6x6, respectively, while generating a pooled representation of them for different locations.

A convolutional layer of 1x1 is applied to the output to reduce the dimension of context representation. With bilinear interpolation, these low-dimension feature maps are directly upscaled to correspond to the original feature map in size. The pyramid pooling module generates different levels of features that are subsequently combined into a singular feature map. This is then passed through a convolutional layer to yield the final output. [61].

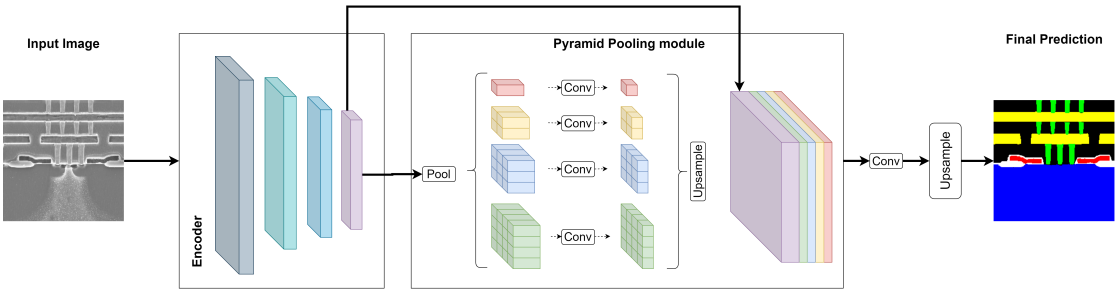


Figure 2.17: PSPNet architecture. It relies on a multi-scale analysis, enabling it to extract features at different scales and capture rich context information. This is done via a pyramid pooling module in which the features are fused under four different pyramid scales: 1x1 (red), 2x2 (yellow), 3x3 (blue), and 6x6 (green).

The Feature Pyramid Network

The Feature Pyramid Network (FPN) operates according to the same top-down principle with skip connections as U-Net and SharpMask [62]. In contrast to models that predict only at the last stage, FPN predicts at each stage, thus combining semantically robust low-level features with semantically weaker high-level features [63]. In addition to the top-down and bottom-up pathways, lateral connections are incorporated into the network, as shown in 2.18.

- Bottom-up pathway: Each stage corresponds to an individual pyramid level. In general, the most profound layers possess the most robust features. The output of each stage is used to create a set of feature maps, which serve as a reference.

- Top-down pathway and lateral connections: Spatially coarser feature maps are semantically stronger and are upsampled by a factor of two. After passing a 1x1 convolution, the feature space is enhanced by the element-wise addition of feature maps of the corresponding level from the bottom-up pathway.

Panoptic FPN is a variation of FPN that combines instance segmentation and semantic segmentation [64]. It adds a semantic segmentation branch to merge the information from all levels of the FPN pyramids into a single output.

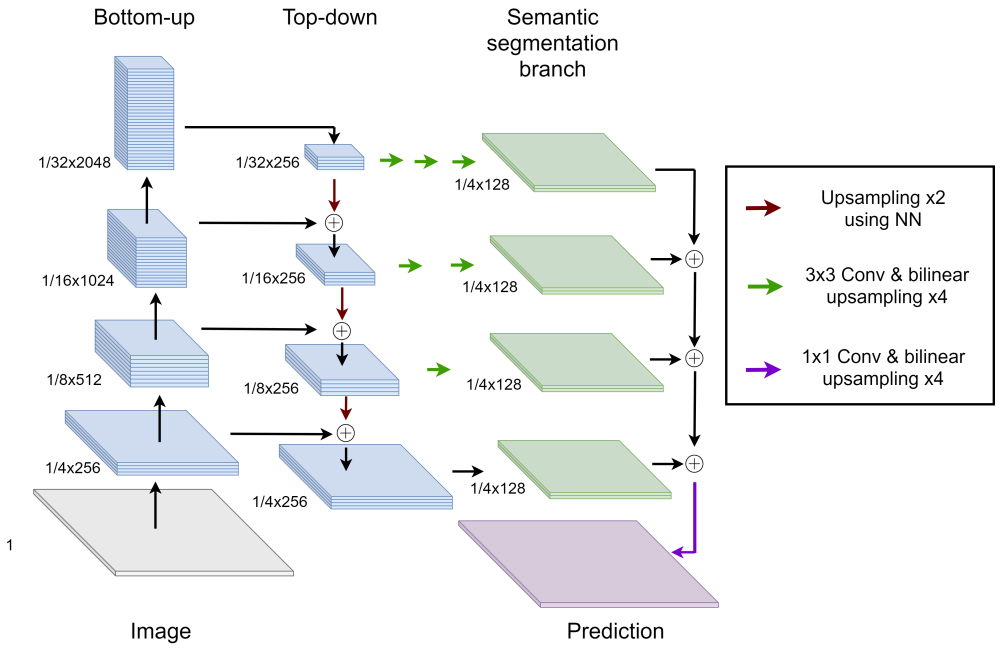


Figure 2.18: FPN for semantic segmentation. FPN consists of a bottom-up pathway (standard network with features at multiple spatial resolutions) and a top-down pathway. The top-down pathway progressively upsamples the deepest pyramid level of the network while adding transformed versions of higher-resolution features from the bottom-up pathway. A separate branch is used to adapt the model to semantic segmentation. It merges the information from all levels of the FPN pyramids into a single output.

Gated Shape CNN

A Gated Shape Convolutional Neural Network (GSCNN) for semantic segmentation is an approach that leverages shape information for improved segmentation results [65]. It proposes a two-stream architecture rather than the traditional method of segmenting images in which color, shape, and texture information are processed in a single deep CNN. An explicit shape stream runs parallel to the feature stream, referred to as the regular stream, and allows leveraging shape information.

Gated convolutional layers (GCL) are an essential component of this architecture, which connect intermediate layers from both streams, allowing information to flow from the regular stream to the shape stream. By filtering out unnecessary information, the GCL allows the shape stream to only process relevant data, which is achieved by only using the features of the regular stream to generate an attention map. Deactivating areas without critical boundary information is achieved by applying the element-wise product of the intermediate representation of the shape stream and the attention map. Through atrous spatial pyramid pooling, both streams of information are combined in a multi-scale manner.

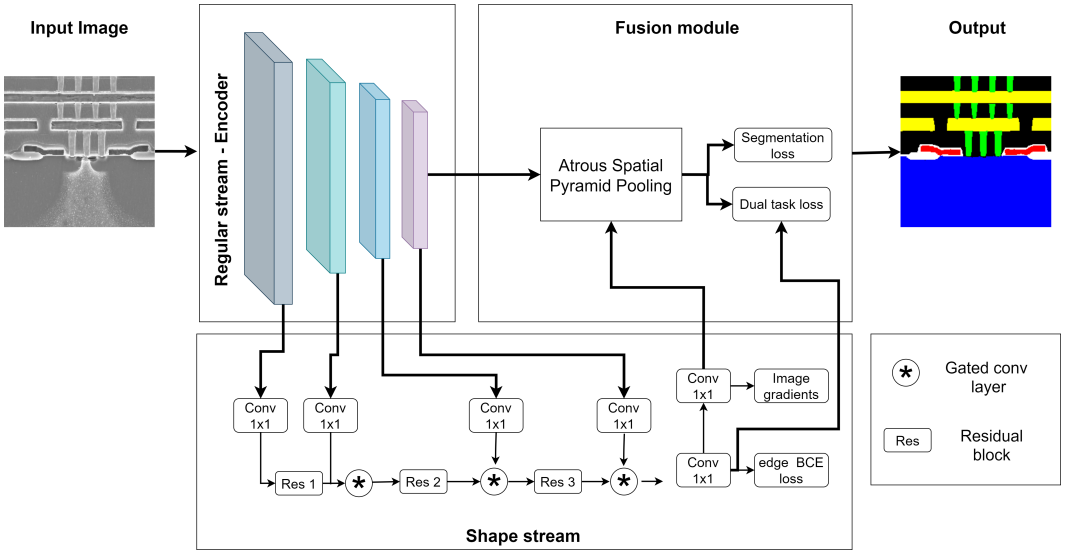


Figure 2.19: GSCNN features a two-stream architecture with a regular stream and a shape stream that explicitly models boundaries.

The resulting segmentation and boundary maps are jointly supervised, and various feature extractors can be used for the regular stream.

2.6 Knowledge Graphs

Basic Concept

Inspired by human problem-solving approaches, knowledge representation aims to enable complex problem-solving for intelligent systems. AI systems are able to leverage human knowledge through such representations, and there have been significant research efforts in academia and industry over the past years [66].

A knowledge graph represents knowledge in a structured way, including entities, relationships among entities, general facts, and semantic descriptions. Various real-world applications are based on knowledge graphs, with Google's Knowledge Graph being a well-known example [67].

Throughout this work, knowledge graphs are leveraged for knowledge-aware models for sensor fusion, object detection, and classification. Additionally, this work intends to leverage previously acquired knowledge and carry out knowledge acquisition to facilitate the implementation of the use cases. Consequently, knowledge graph completion is not in the scope of the present work, rather, the focus is set on relation extraction and entity discovery [68].

Knowledge Representation

The predominant application of knowledge graphs in this work is entity classification and the representation of the relationship between these entities. Human expert knowledge

is used to determine a starting point in a graph from which a directional sub-graph can be derived. An example of such a basic directed graph is to derive the relationship of the entity car to the generic definition of the entity thing. Leveraging the Google knowledge graph API [69], a very basic directional knowledge graph, as shown in 2.20, can be extracted from the Google knowledge graph.

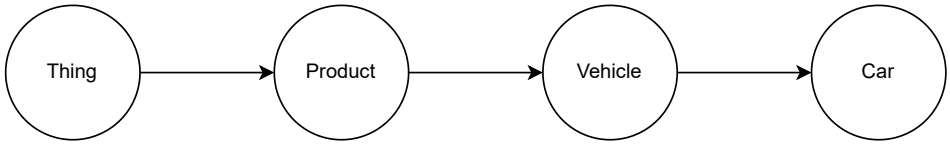


Figure 2.20: Extracted directional knowledge graph: From thing to car

Determining the relationships of objects in the surrounding is essential for automated driving and decision-making. The complexity of objects in the environment and their relationship with each other is highly complex. Depending on a driving situation, the level of detail required to safely resolve the situation varies. In the example shown in figure 2.20, the level of detail correlates with the knowledge graph depth.

When manufacturing ICs, a high degree of automation and monitoring is employed in the production process. This results in well-known boundary conditions regarding the relationship between the components. It is crucial to make sure these conditions are true. Otherwise, the functionality of the IC would be impaired. Figure 2.21 shows a directional graph demonstrating how different components may be connected.

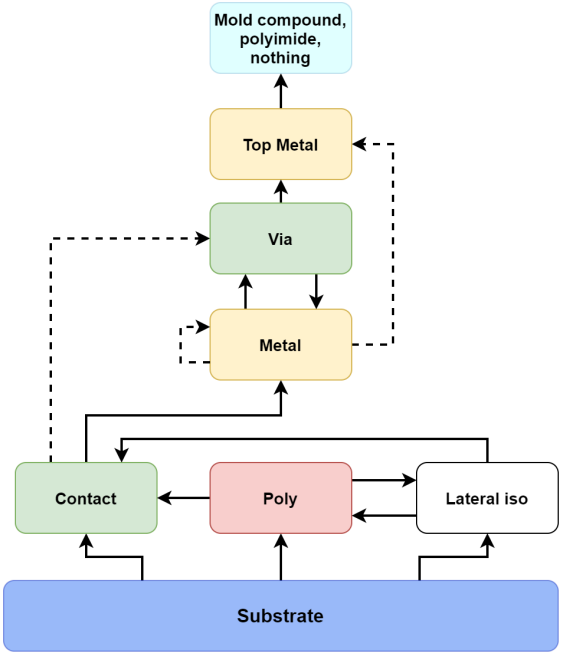


Figure 2.21: Possible connections per class. The arrows indicate the possible connection on top. The colors indicate similar material.

Throughout this work, automotive use cases and industrial use cases leveraging knowledge graphs will be discussed. The used graphs are either based on expert domain knowledge or existing graphs, like the Google knowledge graph. The knowledge is leveraged to simplify the training process by setting boundary conditions and reducing the number of required training samples. Additionally, cascades of classifiers addressing specific use cases are directly derived from knowledge graphs.

3 Methodology

This thesis proposes a robust and explainable framework for leveraging and integrating human domain expertise into ML applications by utilizing data handling techniques and combining established methods, such as Bayesian Networks, with advanced NN architectures. The innovative approach addresses complex challenges in domains such as autonomous driving, quality control in manufacturing, and semiconductor production. Figure 3.1 illustrates the framework and its components. It demonstrates how specific methods are applied to different use cases to facilitate the transfer of knowledge from humans to machines. At the base of this framework is the domain knowledge representation, which underpins all methods and use cases, ensuring that the knowledge applied is relevant and useful.

This chapter outlines the overarching methodology framework used throughout this thesis to integrate domain knowledge into various ML networks across different use cases. The proposed framework systematically combines established soft computing methods with newer NN architectures, implicitly and explicitly incorporating human expert knowledge.

To provide a detailed understanding of how this framework is applied in practice, the following provides an overview of three critical component groups: data handling and integration, model-based sensing, and the hybrid approach combining Bayesian Networks, NNs, and knowledge graphs. The sections in this chapter then dive deeper into each group and explore specific techniques and strategies employed, demonstrating their practical benefits and the enhancements they offer for ML applications.

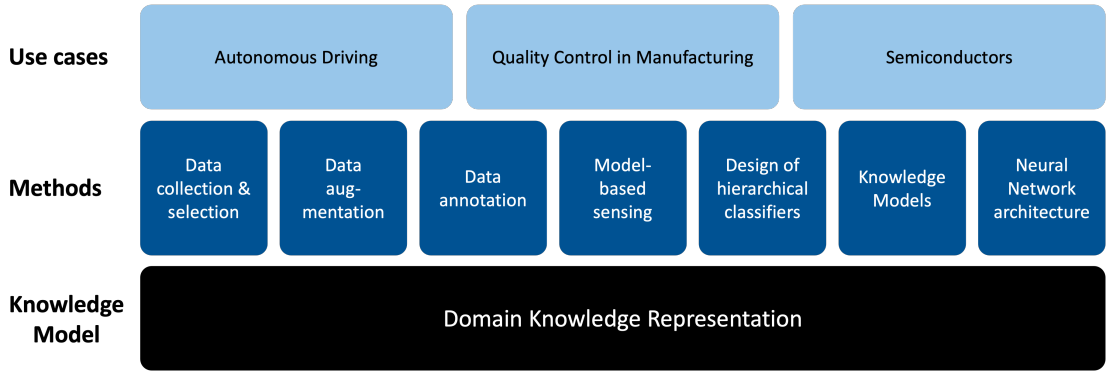


Figure 3.1: Domain Knowledge Exploitation Framework

Data Handling and Integration

The performance and effectiveness of ML-based applications strongly depend on the quality of the training data provided. Data handling is crucial, as the quality and representativeness of data directly impact the resulting models. Thus, the data itself is integral to an ML application. Section 3.1 explains the methods applied in data handling throughout this thesis. These methods effectively incorporate domain knowledge, thereby improving the overall performance and reliability of the systems for the presented use cases. A key contribution of this thesis is the incorporation of domain knowledge across multiple stages of data handling. This approach reduces the need for excessive data collection while addressing the challenges of data variability and annotation.

Model-Based Sensing and Knowledge Integration

This thesis uses the model-based sensing approach as a component in the proposed framework to fuse and process captured data, leveraging preexisting domain knowledge to improve performance and efficiency in various applications. Models of expected objects and scenarios for the investigated use cases simplify the detection of patterns and structures in the captured data. Section 3.2 describes model-based sensing techniques, their benefits, and their applications across different domains.

Hybrid Approach: Bayesian Networks, Neural Networks, and Knowledge Graphs

A key contribution of this thesis is the integration of domain knowledge into ML networks. Section 3.3 highlights how this is achieved through the combination of Bayesian Networks and knowledge graphs. The proposed approach leverages the probabilistic framework for reasoning and uncertainty management while benefiting from the structured representation of data entity relationships in knowledge graphs to enhance inference and reasoning capabilities.

Furthermore, this thesis introduces a combination of Bayesian Networks, NNs, and knowledge graphs as part of the proposed framework. This hybrid approach leverages human expertise effectively, ensuring that the resulting ML models are both accurate and interpretable. The integration of these techniques allows for a comprehensive understanding of the data, improves model performance, and provides robust solutions across various applications, including autonomous driving, quality control in manufacturing, and semiconductor production.

This chapter illustrates how the proposed framework and its components can be effectively applied to real-world problems. Subsequent chapters will explore these methodologies in-depth, demonstrating their practical benefits and the enhancements they offer for ML applications. By systematically incorporating domain knowledge at various stages of development, the aim is to create ML systems that are both highly performant and aligned with the specific needs of their respective fields.

3.1 Data Handling

Data is collected from a variety of different sensors, depending on the use case. It is crucial to understand the use case thoroughly, including how captured data varies depending on external conditions or within the applications.

For autonomous driving, for example, it is important to understand under which environmental conditions the desired sensor setup will be operated. The environment is subject to the dynamics of road users and changing weather or lighting conditions. Depending on the environmental conditions, the captured data varies. This is evident when comparing camera data of a vehicle driving during daylight with a vehicle operated at night during foggy conditions; the appearance of the captured images varies drastically. Similar effects occur for RADAR and LiDAR data under changing environmental conditions.

In other cases, like manufacturing, the sensor set is fixed and static, while the objects to be classified are subject to variances during production. Depending on the quality of the production process, distinct patterns occur that need to be captured so the resulting feature space can be separated to achieve the desired quality assessment.

The variance observed for a desired use case can be approximated by leveraging domain knowledge on the application. In this context, knowledge graphs, as explained in section 2.6, can be applied to understand which variances to expect once the solution is in production. This knowledge is crucial for creating a representative dataset, which forms the basis for training ML applications. Domain expertise allows for the formulation of requirements for a dataset, enabling targeted data preparation. These criteria include completeness with respect to capturing occurrences of probes as expected once deployed. Additionally, diversity and class balance are important features of the data necessary to achieve high-quality outcomes.

The following steps are part of the data preparation:

- Data Collection and Selection
- Data Augmentation
- Synthetic Data Creation
- Data Pre-processing
- Data Annotation

This work proposes directly leveraging domain knowledge throughout all steps, from data collection to processing and annotation, to enable use cases to meet the desired performance level while reducing the need for excessive data collection.

Data Collection and Selection

The methodology for data collection and selection can be meticulously designed to ensure that the captured dataset is representative. This is achieved by emphasizing variety

and diversity throughout the data collection phase. For autonomous driving, the car and its sensor set are tested under various environmental conditions, at different times of day, and in diverse settings such as busy urban areas and highways. Limiting the data collection to the same route under similar environmental conditions results in low overall diversity but a redundant and representative dataset for that route under specific conditions. Both options could be desired, depending on the targeted applications.

In manufacturing, production environments are controlled, and repetitive tasks are performed in a well-controlled environment. This limits the possible occurrences of probes in the dataset, which can be addressed when collecting the data. The aim is to collect a dataset with a high level of completeness, containing a representative set of samples. Examples of the effects of data collection and selection are demonstrated in chapters 6, 7, 8, and 9.

Data Augmentation and Synthetic Data Creation

Once the data collection and selection process is well understood or completed, its limitations become clear. In many cases, collecting a complete set of data that fully represents the use case is not feasible. This directly impacts the quality of any solution built on top of this data. This shortcoming can be addressed by leveraging data augmentation techniques and synthetic data. Both approaches enable the creation of a more complete dataset, resulting in a richer feature space.

There is a large variety of augmentation techniques; a comprehensive overview can be found in [70] and [71]. Whenever augmentation techniques are applied, it is crucial to understand the domain context. Augmentation techniques must be carefully selected to ensure that they generate relevant variations tailored to the specific use case. However, if techniques are applied that result in patterns that do not naturally occur as part of the desired use case, the effect of augmentation on performance can be negative. For example, in semiconductor manufacturing, the process introduces strong constraints on the topology of ICs, as shown in the knowledge graph in figure 2.21. Augmenting the data while adhering to the topology results in a more representative feature set without capturing additional expensive data samples. However, violating these constraints would result in reduced data quality, directly impacting the quality of models based on the data.

Another approach used throughout this thesis to address the data needs of applications is the creation of synthetic data. Many use cases prohibit data collection under certain circumstances or involve events that occur so rarely that capturing them in the desired quantity is not feasible. However, it is crucial that rarely occurring events are part of the training dataset; otherwise, non- or underrepresented classes will not be identified correctly.

Due to the underlying models used for simulation, synthetic data poses the challenge of deviating from data as observed in the real world. This prohibits applications from being exclusively trained on synthetic data and stresses the importance of combining it with real-world data. One approach to address the shortcomings of simulation engines is to capture the physical properties of the environment and create a digital twin that

considers these properties as part of the simulation. A detailed description of leveraging synthetic data and creating a digital twin can be found in chapters 6 and 7.

Data Annotation

The annotation process influences the performance of a machine learning application and the required amount of training data. Human annotators need to follow guidelines to achieve a homogeneous annotation outcome. This requires a well-structured workflow followed by different individuals. Especially when handling ambiguities during annotation, it is important to resolve them based on clear guidelines to avoid inconsistencies in the data, which would negatively impact the training process of a NN. This is generally referred to as inter-annotator agreement and requires that the annotation process is strictly followed and that quality measures are in place to verify the validity of the annotations. More details on the impact of dataset biases are given in [72].

Additionally, the choice of annotation approach is critical for training reliable models. Many CNNs are trained based on image data where objects are approximated via bounding boxes. However, these rectangular bounding boxes contain the object but also include many artifacts that are not part of the actual object. One way to address this, especially in the context of segmentation, is to annotate an object via a polygon. This process is more time-consuming and, therefore, more expensive. Another approach is to annotate data on a per-pixel level, which requires excessive resources. Generally, a coarser annotation is acceptable in cases where enough training data is available. However, when this is not the case, a more accurate annotation is crucial to ensure the desired level of performance for the ML application.

The choice of annotation technique is, therefore, strongly driven by the use case and the respective domain. The impact of wrongly annotated data on the success of the overall application varies strongly. Consequently, domain knowledge of the availability of data and the implications of the annotation process on the training process are crucial. In chapter 5, two granularities of annotation are combined to enable automotive sensor fusion. Chapter 9 addresses the challenge of automatically segmenting images of ICs, for which training data is expensive to generate and, therefore, sparse. A pixel-accurate annotation is applied to leverage the available dataset in the best possible way.

3.2 Model-Based Sensing

The use cases discussed in this thesis are clearly explained, and knowledge is represented in various ways. One effective method of utilizing domain expertise is through model-based sensing approaches. These approaches use models of expected objects to interpret captured data. By incorporating prior knowledge into data processing, these methods generally enhance efficiency and performance for the specific application.

Introduction to Model-Based Sensing

Predefined models are leveraged to interpret and process data based on expected objects or scenarios when using model-based sensing. This approach is advantageous because it allows domain-specific knowledge to be integrated directly into the application, thereby improving efficiency and accuracy compared to purely data-driven methods.

Domain Knowledge Integration

An illustrative example of domain knowledge on object appearance is provided in chapter 5, where the shape of humans is leveraged. There are various techniques to explicitly or implicitly represent shapes. A detailed overview, including the benefits and shortcomings of these different representations, can be found in [73].

The methodological steps involved in implementing model-based sensing include building and validating models based on domain expertise, integrating these models with sensor data, and continuously refining the models as new data is collected. This ensures that the models remain accurate and relevant for the targeted applications.

Chapter 4 combines the concept of Bayesian Networks with prior domain knowledge of sensor sets. The proposed approach incorporates expertise on sensor sets and, therefore, the combination of sensor data from different sensor modalities to fuse data while handling uncertainty.

The approach integrates domain knowledge to utilize human expertise to improve data interpretation and overall application performance. This integration converts expert knowledge into a formal representation that allows steering data processing and model inference, enhancing the reliability and effectiveness of the applications.

Applications and Broader Relevance

In addition to the specific use cases discussed throughout this thesis, model-based sensing has potential applications in other fields, such as medical imaging and robotics. This illustrates the broader relevance and adaptability of the approach.

3.3 Knowledge Integration in Machine Learning Networks

Throughout this thesis, domain knowledge is integrated into various ML networks for different use cases. The following are the primary networks used by the proposed framework:

- Bayesian Networks for probabilistic reasoning
- Neural Networks based on human-annotated data
- Knowledge graphs for explicit domain representation

Bayesian Networks for Probabilistic Reasoning

One effective technique to reflect human knowledge is the use of Bayesian Networks, which are well-suited for probabilistic reasoning and decision-making. Their ability to handle uncertainty and provide interpretability is crucial for leveraging domain expertise. A detailed explanation of Bayesian Networks is provided in chapter 2, including the rationale for their selection in specific use cases.

Neural Networks and Implicit Knowledge Integration

The architectures for NNs introduced in chapter 2 do not explicitly incorporate domain knowledge. Instead, this integration is achieved implicitly through human-annotated training datasets. Supervised learning relies heavily on data preparation, and the performance of the resulting models is significantly impacted by the quality of this data, as outlined in this chapter. Implicit knowledge integration involves using human-annotated data to train NNs, where the domain expertise is embedded within the data itself. Explicit knowledge integration, on the other hand, involves directly incorporating domain knowledge into the model architecture, such as through knowledge graphs or Bayesian Networks.

Knowledge Graphs for Explicit Knowledge Integration

In contrast to NNs, knowledge graphs explicitly represent human knowledge in a structured way. A detailed explanation of knowledge graphs is given in Section 2.6. Knowledge graphs are leveraged in various ways throughout this thesis. They represent the relationships between data entities and facilitate inference and reasoning in fusion and classification tasks when combined with other techniques, such as NNs.

Integration of Methods

A key contribution of this thesis is the combination of knowledge graphs, Bayesian Networks, and NNs. This integration effectively leverages human domain expertise within the overall framework, as shown in Figure 3.1, while ensuring explainability. Various examples, ranging from autonomous driving and quality control in manufacturing to reverse engineering of semiconductors, are presented in the following chapters.

Application of the Proposed Framework to Use-Case

For instance, human-annotated datasets provide crucial information on road users and the vehicle environment in the autonomous driving use case. This annotated data implicitly integrates domain knowledge into the model training process, enhancing the ability to classify and interpret real-world scenarios accurately.

Chapter 9 outlines how a knowledge graph representing deep domain expertise is leveraged to adapt the NN architecture while allowing for regularization during the

training process of the NN. This approach is an innovative way of incorporating domain expertise by combining the two well-established concepts of knowledge graphs and NNs.

The following chapters present detailed examples of how this methodology is applied to different use cases. Each example demonstrates the successful application of the integrated framework, showing the practical benefits of combining Bayesian Networks, Knowledge Graphs, and Neural Networks.

This methodology chapter outlines a novel approach to integrating domain knowledge into ML models, and the thesis systematically applies the approach to selected applications. The proposed framework addresses the complex requirements of various real-world applications by combining established soft computing methods with modern neural network architectures and leveraging human expertise.

4 Low Latency and Low-Level Sensor Fusion

This chapter presents a stateless probabilistic sensor fusion approach for low-level automotive sensing with data from LiDAR, RADAR, and cameras. To reduce latency in object detection and generate object existence hypotheses for each frame, the method operates directly on the associated data from all sensor modalities without using tracking. It provides a significant advantage over traditional approaches that rely on tracking, as it enables faster and more accurate detection of objects.

Using a combination of overlap and distance metrics, probabilistic fusion incorporates input from both 3D and 2D space, eliminating the need for sensor synchronization. Consequently, the data from the different sensors can be more robustly and efficiently fused. A Bayesian network is used to implement the fusion process, a widely used and well-established approach to probabilistic reasoning.

The proposed approach is compared to a state-of-the-art fusion system that uses multiple sensors of the same modality and relies on tracking for object detection. In an urban environment, low-level sensor data was collected to evaluate the performance of the proposed approach, and the results demonstrate a reduction in the latency associated with object detection. Additionally, the proposed approach was found to be more robust and efficient than the current state-of-the-art system in that it was capable of fusing data from multiple sensor modalities without requiring synchronization. The approach and the results reported in this section have been partly published in [74].

4.1 Introduction

Every year, traffic accidents claim the lives of more than 1.25 million people and result in approximately 20 million serious injuries. Human error is the main cause of these accidents, accounting for over 90% of cases, according to research [11]. In response to this problem, governments and industries have made significant efforts over the past decade to reduce the number and severity of traffic accidents. ADAS systems have been developed due to these efforts, which help to alert and support drivers, decreasing the number and severity of accidents [12]. These figures relate specifically to traffic accidents, but they highlight the significant impact that automation can have on society.

Perception of the environment with low latency is a key challenge for any automated or autonomous vehicle. Reliable and low-latency object detection is crucial for this task and can be achieved by fusing data from different sensors and sensor modalities.

There are various approaches to sensor fusion, including using smart sensors with built-in computational capabilities and signal processing to detect objects. This is currently

state-of-the-art for ADAS systems [12]. However, processing all available information is important to increase detection confidence and reduce latency. Smart sensors do not consider information from other sensors when filtering data and may also rely on object tracking for object detection, which can increase latency.

In the context of a road vehicle environment, a low-level sensor fusion approach is presented that reduces detection delays compared to smart sensors. The approach uses LiDAR, RADAR, and camera data at a low level, enabling object detection based on all available information while decreasing detection delay. A combined temporal and spatial association method is proposed and evaluated for sensor setups that are not synchronized in time. Moreover, a probabilistic fusion network that processes the associated data is described and evaluated. Using test data captured in urban environments under various weather and lighting conditions, the performance of the proposed approach is compared with a commercial state-of-the-art LiDAR-based sensor fusion solution.

4.2 Basic Concepts and Related Work

Object Detection

Due to the emergence of both single-stage and two-stage detector architectures in recent years, substantial progress has been made in the field of object detection. Single-stage detectors, exemplified by YOLO [46] and SSD [45], do not make the distinction between object detection and classification tasks. Consequently, their accuracy is typically inferior to that of their two-stage counterparts. Two-stage networks exhibit greater complexity, entailing an initial object detection or proposal phase, followed by a subsequent classification stage. R-CNN [47] pioneered the two-stage detector paradigm and has since undergone numerous refinements, such as Fast R-CNN [48], Faster R-CNN [49], and Mask R-CNN [50].

Low-Level Sensor Fusion

Integrating data from multiple sources, known as sensor fusion, generally encompasses three primary stages: spatiotemporal registration, alignment, and data-to-target association, culminating in state estimation and prediction [35]. The granularity of the fusion is determined by the extent of the available information, which may range from raw, unprocessed sensor data to high-level representations of objects. Low-level sensor fusion has been a vibrant research area for several decades, particularly in the domain of autonomous robotics [75]. A diverse range of methodologies has been proposed, such as fundamental mathematical principles [35] and fuzzy logic [76]. The use of fusion techniques in embedded systems can improve bandwidth constraints, reduce latency, preserve patterns, and improve information acquisition.

4.3 Proposed Approach

Sensor Specific Pre-Processing

The processing, association, and fusion of sensor data involve several preliminary steps. These steps, detailed below, pertain to data preprocessing prior to its association and fusion and are tailored to the specific sensor employed.

Camera Processing

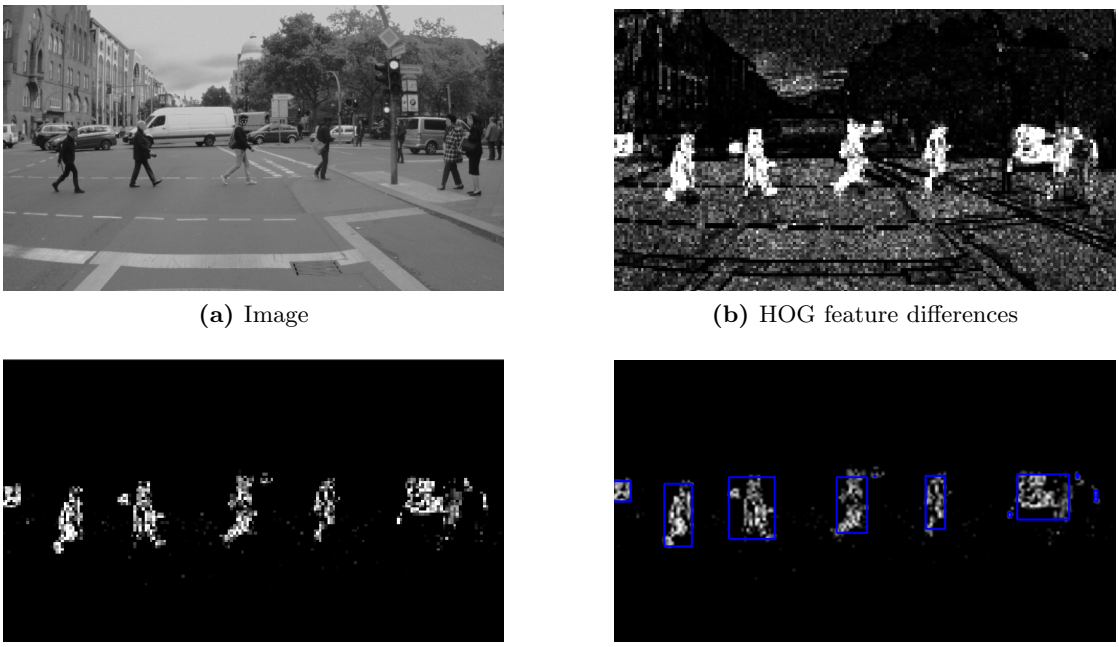
The investigation in this chapter is conducted using a global shutter monochrome camera. While numerous image processing techniques and applications exist, this chapter focuses on utilizing HOG features to demonstrate the benefits of the proposed low-level sensor fusion concept incorporating LiDAR and RADAR. Camera data undergoes preprocessing and exemplifies the potential for integration with other sensor modalities.

The HOG features and their implementation is comprehensively discussed in [77]. A crucial aspect of the proposed methodology is identifying vulnerable road users and vehicles. Although parameterization has not been optimized, the camera output serves to demonstrate the feasibility of incorporating camera data with additional sensor modalities.

It is important to emphasize that HOG features are not directly used for object detection in this work. Instead, the eroded difference between HOG features of consecutive image frames is utilized as input for the fusion network. These differences are clustered, and a bounding box is computed for each cluster. This technique is referred to as HOG frame-to-frame difference (F2FD) throughout this work. This method exhibits the limitation of being unable to detect stationary objects when the vehicle is not in motion. Object detection latency is less crucial when the vehicle is stationary, and detection relies exclusively on LiDAR and RADAR data. The camera-based object detection can transition to a neural network-based object detector to enhance this approach, as demonstrated in 7.

LiDAR Processing

Measurements obtained from LiDAR are clustered for subsequent processing, employing the density-based clustering algorithm DBSCAN [78]. It was selected because of the algorithm's capability for parallelization with large data volumes, its deterministic characteristics, its resilience to noise and outliers, and its ability to manage an unspecified number of clusters. A minimum number of points $MinPts = 4$ is determined according to the parameterization proposed in [78] to minimize the loss of object detection-related information prior to the association phase.
Generating 3D bounding boxes enables an efficient association process for each cluster.



(a) Image

(b) HOG feature differences

(c) Differences after Erosion

(d) Clustered eroded differences

Figure 4.1: HOG F2FD Implementation: In (a), the raw camera image is shown. (b) depicts the HOG feature differences without any further processing. In (c), the HOG F2FD after erosion is depicted. Figure (d) depicts the bounding boxes, which are based on clusters resulting from the DBSCAN used on the data depicted in (c). This figure by Pollach et al. is based on the original publication [74].

RADAR Processing

To produce a list of targets, a peak detection algorithm is applied to the 2D Fourier transformed signal from the RADAR sensor [79]. Each target is defined by a SNR, a radial velocity vector, a distance, and an angle measurement. To group targets originating from the same physical object, RADAR targets are initially clustered. Given the substantial differences between RADAR targets and LiDAR point clouds, it is essential to adjust clustering parameters accordingly. In this chapter, the parameters are set to $MinPts = 1$ and $epsilon = 2.75$ for RADAR. Clusters containing a single target are referred to as individual targets. Clusters are associated with the mean SNR of all measurements within the cluster.

In certain instances, it may not be feasible to group RADAR targets, as only one target may be present per object. Additionally, the sensor might detect ghost targets that do not correspond to real-world objects [33]. Both individual targets and clusters are forwarded to the association phase for further processing.

Proposed Association Method

The proposed method differs from many existing state-of-the-art association techniques in that it employs real-time fusion prior to tracking. Spatial and temporal alignment is used to associate data from multiple asynchronous sensors. The input consists of measurements from individual sensors that have been previously associated with a bounding box.

Each bounding box encompasses an area of A_n. For the 3D scenario, a similar simplification can be applied to the given use case, as all objects traverse on the ground. Accordingly, utilizing an occupancy grid-based fusion approach, 3D bounding boxes are transformed into 2D bounding boxes for the purpose of the association, thereby enhancing efficiency and preserving relevant information. The overlap ratio $overlap_k$ of two bounding boxes, denoted as $\triangle A$, is calculated as follows [74]:

$$overlap = \frac{\triangle A}{min\,(A_1, A_2)} \tag{4.1}$$

Moreover, the distance $distance_k$ between the geometric centers C_n of the bounding boxes is computed in relation to the dimensions of the bounding boxes. Based on the width and length of a bounding box, the diagonal expansion $diag_n$ is determined.

To compute the distance $distance_k$ between two centers, the subsequent equation is employed [74]:

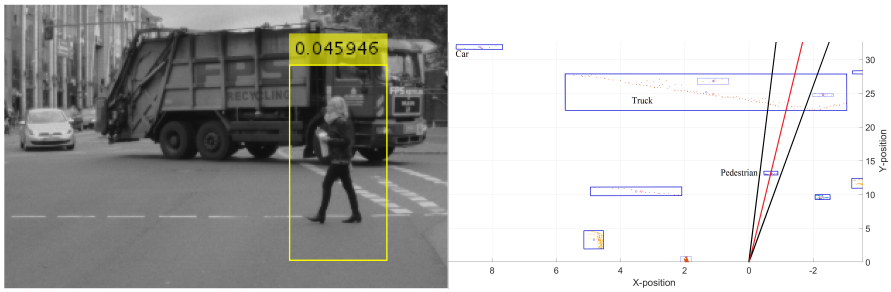
$$distance_k = \frac{d\,(C_1, C_2)}{max\,(diag_1, diag_2)} \tag{4.2}$$

During the association process, it is crucial to consider not only the geometric relationship but also the age of the data. Data association is conducted based on the fastest sensor, which introduces a time difference of t_{fast} between measurement frames at a frequency of f_{max}. Measurements from other sensors are accessible at a frequency of f_{Sensor_n} and have a time difference between frames of t_{delay}. The threshold for association decreases with the increasing age of data available to be associated with the most recent incoming data. This allows for a more tolerant association between sensor measurements of the same object when addressing the movement of dynamic objects and the ego vehicle. The following equation calculates association thresholds based on $overlap_k$ and $distance_k$ [74].

$$t_{overlap_k} = \alpha \cdot \frac{t_{fast}}{t_{delay}} \frac{\triangle A_k}{min\,(A_1, A_2)} \tag{4.3}$$

$$t_{dist_k} = \beta \cdot \frac{t_{delay}}{t_{fast}} \frac{d\,(C_1, C_2)}{max\,(diag_1, diag_2)} \tag{4.4}$$

Data is associated if the distance between the centers of two bounding boxes is below the threshold t_{dist_k} or if the overlap ratio surpasses the threshold $t_{overlap_k}$. In this instance, a threshold is established by adjusting the variables α and β. During this procedure, the fastest sensor serves as the basis for association. The system examines additional data from other sensors whenever the fastest sensor acquires a new set of



(a) Detected Objects in 2D Plane **(b)** Orthographical View

Figure 4.2: Association of 2D and 3D Data: In (a), a bounding box is displayed. The corresponding bounded 3D LiDAR data is visualized in (b) and demonstrates the resulting association using the projected camera rays in the orthographic perspective. The camera rays associated with the left and right edges of the bounding box are depicted in black, while the ray corresponding to the center of the bounding box is shown in red. Only data originating from the pedestrian is associated. This figure by Pollach et al. is based on the original publication [74].

measurements. The worst-case association error, which evaluates the uncertainty level associated with an object's location, can be computed based on individual sensors' known frequencies and latencies. This assumes that objects have a limit to how fast they can move, a concept that is consistent with established physical laws. The size of the estimated object fluctuates with the age of the data in this proposed association technique. However, it minimizes the impact of incorrect measurement assignments attributed to the age of the data [74].

The proposed low-level sensor fusion approach for object detection in a road vehicle environment transforms image data into 3D space utilizing the same model as a 3D to 2D projection. For the conversion from 2D to 3D data space to be executed, two conditions must be fulfilled: the occurrence of an event in 3D space and the event being situated within the FOV of the camera. Based on the camera pinhole model, the algorithm calculates the azimuth angle of the bounding box coordinates. When observing 3D data from an orthographic perspective, the azimuth angles are used to associate camera data with the nearest detected object in 3D space. Any other objects within the determined azimuth angle are not associated with camera data.

Proposed Fusion Network

Statistical inference entails drawing conclusions about the properties of underlying distributions and their parameters based on data analysis. Various approaches and schools of thought exist in statistical inference, including those discussed in [36]. The representation of probabilistic knowledge is often numerical, but graphical representations may be more intuitive for a human to comprehend. A graph structure, with nodes symbolizing propositional variables and edges representing dependencies, is a prevalent model for joint probability distributions. Dependencies and independencies can be depicted using

directed or undirected graphs, respectively. Bayesian and Markov networks were examined as popular methods for handling uncertainty in the context of object detection. Factor graphs integrating these approaches were also explored [80]. An advantage of the Bayesian approach is that it captures conditional independencies. Further comparisons of Bayesian and Markov networks are provided in [36] and [38]. These analyses indicate that a Bayesian network is the most appropriate choice for the desired fusion network, as detailed in chapter 2. As a deterministic and straightforward inference method, the exact inference was selected for evaluation purposes.

The proposed sensor fusion approach for embedded hardware aims to operate in real-time and employs a fusion network. There are both exact and approximate implementations of Bayesian networks that are efficient for inference. A Bayesian network is implemented using a tree structure, with leaf nodes selected based on expert knowledge. The probability of detecting an object increases when data from multiple sources is available. In this instance, the Bayesian network has three sensor modalities contributing to the existence probability hypothesis: LiDAR, camera, and RADAR. The design of the network allows for the integration of additional sensors of various modalities. Expanding the second layer of nodes by incorporating additional parameters and sensors is also possible. Moreover, the network enables the combination of low-level sensor data and object-level data if smart sensors are integrated into the sensor set.

LiDAR data is represented by LiDAR clusters at the leaf nodes of the Bayesian network. During the DBSCAN process, it is assumed that noise has been eliminated. For RADAR data, the leaf nodes are designated as single RADAR targets, RADAR target clusters, and RADAR SNR. The SNR is employed to assess the RADAR data quality and is normalized based on the maximum expected SNR value for the specific RADAR in use. For the camera, two leaf nodes are chosen: one representing the camera SNR, which depends on the magnitude of an F2FD cluster and the noise in the corresponding F2FD data frame, and the other representing the HOG node, which depends on the size of the detected target and the SNR.

Computing the existence probability of the root node is implemented according to the following steps:

1. Initialize all leaf nodes
2. Compute next higher level node distributions depending on leaf nodes
3. Compute next higher level node distributions depending on previous layer nodes
4. Check if the root node is reached; if not, go to step 3
5. Output root node hypothesis

The low-level sensor fusion methodology initializes leaf nodes by utilizing the output generated during the association phase. A specific object may be associated with information derived from single or multiple sensor modalities. Subsequently, the higher-level layers of the tree are computed based on this input. This results in a probability distribution for each individual sensor modality in the higher-level layer. This distribution

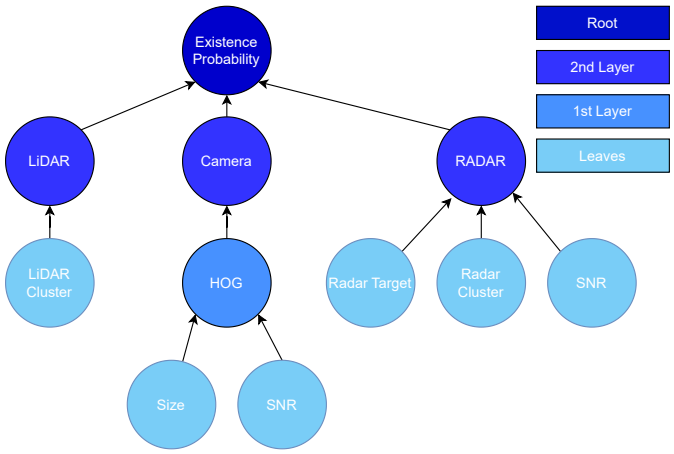


Figure 4.3: Implemented Bayesian Network: The three sensor modalities contribute to the decision of the existence likelihood. The leaf nodes show the information provided to the network.

is used to determine the existence probability at the root node. In instances where leaf nodes exhibit missing values due to the lack of sensor measurements, these are regarded as null values.

4.4 Experiments

Sensor Setup

Data was recorded using a car equipped with LiDAR, a camera, and RADAR sensors in an urban environment. The mounting positions of all devices and the sensor setup have been calibrated in advance, and the extrinsic and intrinsic calibration matrices are known. The vehicle used for data capturing was configured with the following setup [74]:

- **Ibeo Lux LiDAR sensors:** In total, six Ibeo Lux LiDAR sensors are mounted around the car, resulting in a 360 degree coverage of the vehicle environment. Each LiDAR has four laser beams, a horizontal FOV of 110 degrees, and runs at $12.5Hz$.
- **Front RADAR:** This is a short-range RADAR operating at a frequency of $24GHz$ with a FOV of 100. The sensor returns targets at a frequency of $30Hz$.
- **Front camera:** The used camera is a global shutter grayscale camera with a resolution of 752x480 at $25Hz$. The FOV covers 67 degrees.
- **High precision LiDAR sensor:** A Velodyne HDL-64 using 64 beams is mounted, but it is not used in this present work, as it is non-automotive grade. It serves as a reference to determine ground truth.

These algorithms are based on information gathered from LiDAR sensors, RADAR data, and input from the forward-facing camera. Low-Level Sensor Fusion (LLSF) is evaluated in the area where all three sensor modalities intersect, particularly in the FOV of the front-facing camera. It is necessary to constantly monitor the area ahead of the vehicle as it is deemed of utmost importance. Environmental circumstances may include poor visibility, precipitation, or fog. However, the acquired dataset does not include any scenarios involving snowfall, rain, or fog.

Association Implementation

Operating at distinct frequencies, the sensors exhibit temporal discrepancies between their measurements. The camera sensor possesses a frequency of $f_C = 25Hz$, and the RADAR runs at a frequency of $f_R = 30Hz$, and the LiDAR at a frequency of $f_L = 12.5Hz$. The provided datasets encompass information from a single camera and one RADAR unit. The LiDAR serves as the sole sensor offering additional environmental coverage around the vehicle, whereas only the three-front facing LiDARs with an overlapping FOV are used for sensor fusion. In line with the proposed association strategy, constant values are chosen for the time-related parameters that are relevant to the association process. The LiDAR-induced worst-case delay is $t_{delay} = 80ms$. For associating data, allowing a more sensitive threshold or smaller overlaps is reasonable since this leads to the assumption that the object is larger than it actually is. This can be achieved by selecting the maximum sensor frequency, which results in $t_{fast} = 33.3ms$. The worst-case delay assumption is consistently employed throughout the association procedure, and parameters α and β are chosen accordingly.

In the initial phase of the process, 3D bounding boxes are associated, assuming that all relevant objects have two degrees of freedom. In light of the fact that any road user is connected to the ground, this approach is considered viable. The bounding box overlap is determined by executing an orthogonal projection onto the ground plane. Camera data is then associated with LiDAR and RADAR bounding boxes projected onto the ground plane.

Recorded Scenarios

The available dataset covers a wide range of use cases, from urban highways to bustling city intersections. Nevertheless, no data from extreme environmental conditions is available due to the prototypical sensor set installation.

The algorithms need to be capable of processing data from various object types in the environment. The camera resolution limits the maximum distances, which prevents ground truth determinations at greater distances. Consequently, the evaluation focuses on urban use cases, specifically those involving vehicles and pedestrians in close proximity. The proposed sensor fusion methodology was assessed using the following three scenarios:

The busy urban crossing scenario (I) comprises various vehicles, pedestrians, and bicycles. This scene was chosen due to the high volume of road users and the existence

of partially occluded objects. As the test vehicle approaches an intersection, it halts at a red light while other vehicles pass at speeds of approximately 50 km/h and pedestrians cross the street. Distinct viewing angles and occluded road users are observable in this scene. The scenario spans 42 seconds.

The red light scenario (II) involves only passing vehicles as the test vehicle approaches a red light and waits at the stop line. Due to the presence of various vehicle types, this scenario was chosen. A notably small city car, as well as a bus, can be found in this scene. An appropriate evaluation of object detection delay can be performed because a bridge pillar conceals vehicles until they enter the intersection. The scenario's duration is 34 seconds.

During the night scenario (III), lighting conditions are challenging due to twilight. All vehicles have their lights on, and pedestrians occupy the sidewalks. The test vehicle approaches vehicles utilizing two lanes, causing the occlusion of other vehicles. The presence of pedestrians without active illumination from a light source is given. The scenario persists for 40 seconds.

Table 4.1: Scenarios

Scenario	Category	Number	Max distance (m)	Min distance (m)
(I) Urban Crossing				
	Pedestrians	7	45	5
	Vehicles	27	50	15
	Bicycles	2	45	34
(II) Red Light				
	Pedestrians	-	-	-
	Vehicles	10	50	8
	Bicycles	-	-	-
(III) Night				
	Pedestrians	8	25	3
	Vehicles	19	50	2
	Bicycles	-	-	-

Comparison of Fusion Approaches

The proposed LLSF methodology is compared with a state-of-the-art LiDAR-based sensor fusion system, referred to here as the Fusion Box (FB). The unprocessed sensor data received by both systems is identical, and their outcomes are recorded simultaneously. The LLSF only yields object detections when LiDAR data is available to ensure a fair comparison. The FB incorporates data from six LiDAR sensors, of which the three front-facing ones with an overlapping FOV are used. The evaluation relies on scenarios with independently-generated ground truth data for each sensor modality. Data from all sen-

sors is manually annotated to establish ground truth data. The detection performance of pedestrians and vehicles is examined separately.

The LLSF system operates on individual frames without implementing tracking. As soon as the Bayesian network's output exceeds a threshold of 0.5, the object is deemed to have been detected. Contrary to this, the FB employs an object tracker and trajectory prediction, which may provide an advantage in instances where objects are partially or entirely concealed. Although the LLSF system does not include a tracking algorithm, it can easily be integrated if desired.

Three scenarios were selected that represent typical urban use cases at night and during the day. As a result of these scenarios, road users are captured at varying distances, from varying angles, and in challenging lighting conditions. The results for the analyzed scenarios are displayed in Table 4.2.

Table 4.2: Overall Number of Object Detections

Category	Fusion	TP	FN	FP	F1-Score
Vehicles					
	LLSF	5763	48	24	0.9938
	FB	5584	227	32	0.9768
Pedestrians					
	LLSF	865	33	5	0.9786
	FB	852	46	64	0.9394

On average, 94.38% of vehicles and 85.63% of pedestrians are identified by more than one sensor modality. Objects detected in such a manner generally exhibit higher confidence compared to those identified solely by a single sensor modality, which is LiDAR for the current use case. The FB system possesses an average detection delay of $T_{delay} = 176.5ms$ for vehicles and $T_{delay} = 268.28ms$ for pedestrians. In contrast, the LLSF system demonstrates a delay of $T_{delay} = 92.65ms$ for vehicles and $T_{delay} = 105.92ms$ for pedestrians.

Figure 4.4 illustrates the number of objects detected by the distinct fusion techniques for 100 measurement frames of scenario (II) to allow a qualitative assessment.

4.5 Conclusion and Outlook

A LLSF methodology was introduced, leveraging data from LiDAR, RADAR, and camera sensors to enhance object detection prior to tracking. A LLSF approach implements a centralized fusion process that functions on low-level sensor data, thereby reducing detection latency and enabling the use of asynchronous sensor configurations. A clustering strategy was employed to manage low-level sensor data effectively for the purpose of the association. The fusion network is based on expert domain knowledge that is leveraged for the implementation of the Bayesian fusion network.

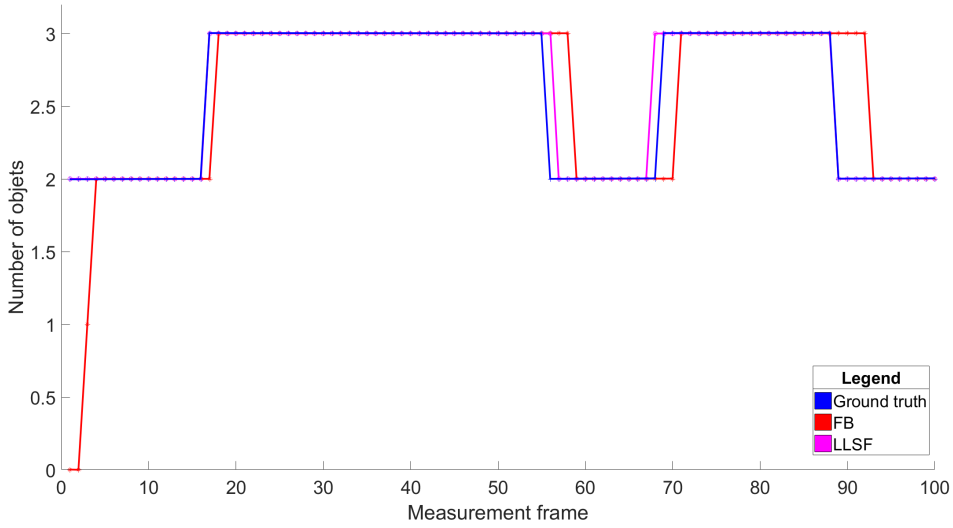


Figure 4.4: Object Detection by Fusion Approaches: The number of objects detected by the two fusion approaches is compared to ground truth. The FB introduces a time delay at the beginning of this example. This figure by Pollach et al. is based on the original publication [74].

Experimental outcomes revealed that the LLSF approach predominantly benefitted vehicle detections in terms of detection latency due to data available from multiple sensors. However, the state-of-the-art FB system, which employs object tracking and trajectory prediction, exhibited superior performance for partially or fully occluded objects while generating a higher number of false positives due to tracker usage. Furthermore, the LLSF approach demonstrated a reduced delay in detecting objects once they enter the field of view of the sensors, allowing quick reactions and the possibility of preventing or mitigating collisions. In response to increasing distances, LiDAR measurements become sparser, causing the FB system to experience increased detection latency or non-detection. In contrast, the LLSF compensates for this effect by using data from alternative sensor modalities.

Future Work

It is anticipated that future research will include optimizing and expanding the Bayesian network used in the LLSF approach and investigating the feasibility of training classifiers directly on multi-modal sensor data. The proposed method for determining the object existing hypothesis has the potential to facilitate the process of object detection and region proposal in two-stage neural networks while also providing a richer feature space than a single modality-based input. The reduced detection latency is crucial for safety-critical applications and one of the key advantages of a LLSF. The proposed approach lays a robust foundation for future work on LLSF systems, and there is potential for

further refinement and enhancement of such systems to yield even better results in future studies.

It is crucial to contemplate the primary conceptual distinctions between the compared sensor fusion systems, encompassing the number of sensor modalities used and the application of object tracking and trajectory prediction. LLSF operates exclusively on single measurement frames and does not implement these methods, while FB does. Consequently, future work will make use of tracking to improve the overall system performance.

5 Deep Multi-Modal Fusion for Object Detection Using Segmentation

5.1 Introduction

In recent years, ML has tremendously impacted the development of automated and autonomous vehicles. Leveraging multiple sensors combined with deep learning approaches has proven essential to cope with the large variety of scenarios a vehicle can encounter while operating under changing environmental conditions with highly dynamic objects [81]. Advancements in hardware technology have made training deep convolutional neural networks, utilizing massively chained backpropagation, computationally feasible [82, 83]. Thus, introducing significant depth to CNNs has allowed for the necessary complexity to approximate highly non-linear problems. Deep models have been established as state-of-the-art methods in object detection [47] and semantic segmentation [50]. Image processing networks for those tasks have been a strong research focus, while RADAR and LiDAR-based object detection has improved in recent years. All sensor modalities are required to achieve a high level of automation for coping with a large variety of environmental conditions. However, the processing of the different modalities is mostly separate, and fusion happens at a high level, as explained in chapter 2. Understanding at which level data from different sensor modalities can be fused most effectively is a crucial question and impacts the design of a neural network. Sensor fusion at different levels within a deep neural network is essential for improving multi-modal-based object detection and semantic segmentation. Leveraging human domain knowledge offers the opportunity to reduce the complexity of these tasks.

5.2 Related Work

Shortcut Connections

Due to increasing capacity, deeper networks should theoretically produce better results. However, it has become apparent that they also become more intricate to train [84]. One major issue with depth is that of vanishing or exploding gradients [85]. Increasing depth leads to the multiplication of ever smaller values during backpropagation. This, in turn, results in infinitesimal weight updates, preventing the model from converging, i.e., rendering training unsuccessful. The problem has been countered well by normalized weight initialization [86] and layer-wise output normalization [87]. The success of these techniques has further accelerated the establishment of even deeper networks while improving performance [88, 84, 46, 89]. While performance was expected to simply in-

crease as a function of depth, at a certain level, it declined. The authors of ResNet first addressed this issue successfully on a big scale and set new standards for the state-of-the-art in deep networks. Their solution consists of residual functions, i.e., functions that simply allow learning to skip parts of the network in case performance decreases [84]. Following this work, different kinds of short-cut connections have been used to improve performance with increasing depth while maintaining the ability to optimize [90, 89, 88].

Autonomous Driving

These improvements in computer vision methods have been vital for the research community to attempt an implementation of AD [81]. To responsibly realize autonomously operating vehicles, any deployed model must generalize to a wide variety of scenarios, allowing the model to cope with situations ranging from suddenly appearing pedestrians in city traffic to correctly estimating truck movements on highways. Large-scale datasets are a key requirement in this undertaking. The respective datasets often contain data from RGB cameras as well as LiDAR sensors [91, 92, 93, 94]. The influence of different fusion mechanisms has not yet been widely studied in the context of deep models.

Deep Multi-Modal Fusion

In highly dynamic scenarios, relying on multiple sensors has two advantages. Redundancy makes systems more robust against disturbances, while relevant supplementary information allows for performance improvements in the absence of failure or external impairments. In AD, adding depth information on top of the 2D RGB camera images has proven particularly invaluable [81]. While there is a rich body of literature assessing fusion for traditional computer vision algorithms, the questions of 'What and When to fuse' in AD applications have been identified as two of the yet-to-be-conclusively answered challenges regarding deep neural networks [81, 95]. The impact of the processing level at which multi-modal data, specifically LiDAR and RGB data, is fused is systematically explored in this chapter. Approaches found in the literature are explained in chapter 2 and range from early, mid, and late fusion [96, 97, 98, 99, 100] and continuous fusion [101, 102, 103] to more complex schemes, for example, basing the processing of one modality entirely on the intermediate results stemming from another modality [104]. All these efforts lack comparability due to their widely varying architectures, optimization procedures, and employed datasets. Concerning the representation of depth data, as, for example, given by LiDAR recordings in most current AD datasets [91, 92, 93, 94], there are currently mainly three approaches the scientific community utilizes:

- Point clouds hold a lot of information but are hard to process using CNNs. The given space is sparsely populated by data, and the three-dimensional nature makes computations exponentially more expensive compared to their 2D counterparts [105]. This approach projects the depth data into a 2D array containing the data of the third dimension as pixel values. Accordingly, these representations are processable for any standard deep computer vision architecture based on 2D convolutions.

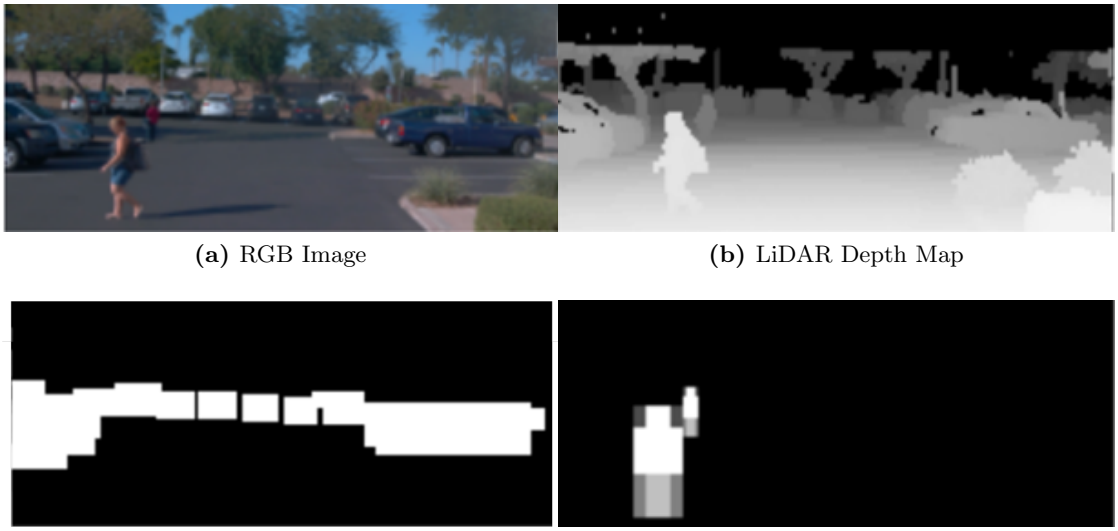
- The birds-eye representation, i.e., the array dimensions are depth and width, leaving the pixels to hold height values, is used in state-of-the-art architectures. This necessitates the development of strategies to correlate the LiDAR data with the perspective-specific RGB data [105]. A representation of this application is exemplified in chapter 4.

- Point-of-view based 2.5D depth maps keep the computational requirements low and allow easy inter-modality linkage of information [105].

Considering the flaws of the first two representations, the 2.5D option is chosen for the proposed model. In this context, a new and computationally lightweight way to derive the depth map representation is proposed, given LiDAR data that is synchronized and projected into the frame of the RGB camera. It is possible to efficiently derive a dense, human-readable depth image from sparse LiDAR data through splatting and pooling.

Semantic Segmentation of Bounding Boxes

Currently, object detection and semantic segmentation differ by the utilized ground truth data. In the field of object detection, bounding boxes are typically utilized. These are generally composed of four coordinates that outline the position and dimensions of an object, in addition to a class label [106, 107, 91, 93]. The process involves performing regression on these bounding boxes. On the one hand, having this representation produced by humans is very cost-efficient, as each object only needs to be roughly approximated by the surrounding quadrilateral. Furthermore, it allows working with an explicit representation in further processing. On the other hand, it is highly inflexible. Imagine a scenario where a person points somewhere: Although someone simply standing, keeping the arms close to the body, would be assigned the same label, the ratio between background and object changes vastly. The background composition captured by the corresponding bounding box might be too little to infer relevant or class-specific context for detection, making properly computing the coordinates unnecessarily complex. In semantic segmentation, in addition to bounding boxes, each pixel is uniquely assigned a class label by a mask [106, 107, 108]. Compared to the bounding boxes, this representation is far more flexible, as only pixels belonging to a certain class are assigned the according label. However, the cost of creating these masks is substantially higher because each object's silhouette must be traced by hand. Due to this, datasets for semantic segmentation are small compared to their object detection equivalents [106, 107, 108, 93]. The proposal is to create heat maps from bounding box labels to leverage the vastness of high-quality, large-scale object detection datasets for pre-training semantic segmentation networks. This suggestion is explored by creating class-wise segmentation masks from bounding box labels. Given the bounding box dimensions, a function is employed to roughly approximate the silhouette of the corresponding object class. As described in more detail in the methods section, a network designed for semantic segmentation on these labels is trained.



(a) RGB Image **(b)** LiDAR Depth Map

(c) Ground Truth: Vehicles **(d)** Ground Truth: Pedestrians

Figure 5.1: Example input and ground truth data

Contribution

This chapter systematically assesses performance differences concerning the degree to which LiDAR and RGB data are pre-processed separately prior to fusion. Accordingly, it is investigated at which level of a network to fuse both data types. Furthermore, a novel approach to object detection is proposed, namely utilizing semantic-segmentation-mask-like heat maps instead of bounding boxes. An approach is proposed that allows the inclusion of domain knowledge on object appearance into networks while simultaneously reducing complexity and improving segmentation and detection quality. An architecture that heavily relies on shortcut connections is proposed to maintain the compactness of a model outputting maps equaling the size of the input. DenseNet [89] and U-Net [90] are combined into a highly compact model with a comparably mediocre number of parameters. The block structure of the encoder simply allows for investigating the effect of joining LiDAR and RGB data at different levels of the network. Like FuseNet [101], dedicated encoding modules are used for RGB and LiDAR data, respectively, followed by a joint decoder. In contrast to FuseNet, the data is variably fused at one point of processing by the network, which allows the assessment of different fusion depths. Moreover, an efficient way to convert the LiDAR data offline into dense point-of-view 2.5D depth maps is introduced by simply allowing processing by any standard 2D-convolution-based network. The Waymo Open dataset [93] is employed, as it delivers high-quality, synchronized data.

5.3 Methodology

Data and Pre-processing

A subset of the Waymo Open dataset for AD [93], consisting of about 20.000 labeled recordings, is used to assess the model. A recording consists of a high-resolution (1280x1920 pixels) RGB image and about 16.000 LiDAR measurements corresponding to the respective image frame. Information about the presence of vehicles, pedestrians, and cyclists is taken from the 2D bounding box labels provided. Traffic signs, which are included in the dataset, are not investigated. The temporal order, as given implicitly through sequences of recordings, i.e., temporally consecutive frames, was neglected and completely removed during the sample randomization process. The dataset is divided into training, validation, and test set (60%:20%:20%), respecting the missing independence of data from the same sequence, i.e., one of the sets exclusively comprises all data from a given sequence. The aim is to evaluate the underlying methodology, not its optimization potential. Consequently, compared to state-of-the-art, no regular data augmentation operations, such as flipping, rotating, or cropping, are used.

In the training set, there is a substantial difference in the occurrence of object classes. On average, 10,86% of image pixels belong to the vehicle class, 0,32% are classified as pedestrians, and only 0,05% are classified as cyclists. As a result of putting these numbers into perspective, it can be concluded that there are 33 times more vehicle pixels than pedestrian pixels and even 208 times more vehicle pixels than cyclist pixels. In the corresponding subset of about 12000 images, approximately 92% of them contain vehicles, 62% contain pedestrians, and only about 8% contain cyclists. Thus, the data imbalance is twofold. As can be seen from the latter statistic, pedestrians and cyclists appear in fewer images than vehicles. Based on these differences and the average number of pixels covered by each class, the imbalance is not solely due to the number of class occurrences but also due to the size of the objects. In comparison to vehicles, pedestrians, and cyclists cover fewer pixels on images.

RGB images are taken from the front camera only to limit the scope while focusing on the most crucial FOV for automated vehicles. To make processing feasible, all images were average pooled with filter size and stride size ten, resulting in images of size 128x192. Average pooling was chosen due to its robustness towards outliers compared to max pooling.

The LiDAR measurements, corresponding to the frame of the front RGB camera, are pre-processed into a 2.5D representation, equivalent to point-of-view depth maps. As proposed in [105], it is argued that the resulting representation enables easy association of both data types by relating them and maintaining simple processing by the same methods, i.e., 2D convolutions and pooling. Firstly, the data is converted such that standard operations, such as max-pooling, are reasonable to apply. Secondly, the data is projected into the depth map format. Finally, the maps are densified. In the following, this three-step pipeline resulting in high-resolution and human-readable depth maps, as shown in figure 5.1, is described in more detail.

1. **Conversion**: Two linear functions were employed to invert the data. One function scales distances between 0-25m to pixel values between 255 and 100. The other function converts distances between 25-75m to values between 100 and 0. Thus, close-range differences were enhanced. Note while the scaling is normalized to match the range of RGB pixels, float precision is preserved. The motivation for converting the data this way is threefold:
   - Small distance differences can mask relevant detections, like moving pedestrians. In close-range scenarios, the granularity and density of distance measurements are most relevant. In addition, the data collection vehicle utilized to collect the data also reflects this rationale. Four of the five sensors only record short-range data, with a 25m cutoff distance. A top-mounted mid-range sensor with a maximum recording distance of 75m is also included. Therefore, short-range differences are enhanced due to the importance of granularity in short-range scenarios. Due to the limited number of sensors and measurements, the sensor data for distances exceeding 25 m is sparse, thereby preventing any higher degree of detail or granularity.
   - The inversion of the signal enables the use of max-pooling. The use of max-pooling after inversion places a greater emphasis on values of close distances.
   - Furthermore, the underlying hypothesis is that higher intensity values, i.e., close distances after inversion, make it easier for the network to learn since the weight value impacts high input values more. Low input values multiplied by any weight result in low values. For the sake of the scope of the present work, this is not assessed.
2. **Projection**: LiDAR distance measurements were projected into a 2D camera frame using the utility toolbox from Waymo. All empty values of the 1280x1920 array containing sparse depth information were set to the maximum recording distance of 75m, which after inversion, results in a value of 0.
3. **Densification**: A 5x5 splatting filter was applied to all values originating from a measurement [109]. Max-pooling with a stride of ten was applied to achieve computational feasibility as well as to match the final RGB image size of 128x192. To achieve a dense result through max-pooling, the kernel size was set to 10x20. Two factors justify this measure: firstly, the horizontal sampling resolution of Waymo's LiDAR is higher than the vertical sampling resolution. Secondly, extending a measurement vertically does not affect the localization of an object negatively in the context of the intended application, while stretching an object horizontally would distort perception considerably. To illustrate this claim, it is suggested that changes occurring orthogonally to the direction of ego-motion become relevant more quickly than movements along the axis of ego-motion. For example, a pedestrian crossing the street in front of the car versus a pedestrian walking parallel to the car. The splatting operation preceding the pooling may be replaced by adjusting the filter size of the pooling kernel. The densification step delivers similar results to nearest-neighbor upsampling while being more lightweight. Given the

constraints of embedded hardware in AD, the computational requirements are of great relevance.

Output

Heat maps were created by converting the labels, which were provided as 2D bounding boxes. It is important to note that segmentation masks differ from heat maps in that all maps are separated, resulting in class-wise heat maps. The generated heat maps have been grossly simplified for the sake of scope. The bounding boxes of vehicles and cyclists have been fully transferred into the segmentation mask format without major changes. A zero value was assigned to pixels in the background, but a value of one was assigned to pixels that overlapped bounding boxes. As a result of qualitative assessment of the bounding boxes and using domain knowledge on the appearance of humans, pedestrian silhouettes are generically approximated using a basic function. This function intends to approximate the shape of a pedestrian. A value of one is assigned to regions of the bounding boxes covering the head and upper body. Other parts of the bounding boxes have a value of .75, .5, and .3 to reflect the likelihood of being part of a pedestrian versus being background. The resulting pedestrian approximation is shown in figure 5.1.

Architecture

As shown in figure 5.2, the network architecture consists of an encoder, a feature extracting path, and an upsampling path, the decoder. A U-Net-like architecture [90] is used, where the encoder is divided into blocks whose intermediate results are fed back through skip connections to the decoder. Through the use of skip connections, all previous intermediate results are fed to each subsequent layer within the same processing block of the encoder, eliminating redundant computation of filters. Thus, it represents features very densely, and the feature extractor performs well despite its modest size. DenseNet [89] consists of four so-called 'DenseBlocks', which allows it to be easily customized for a specific purpose. Respecting the original implementation of DenseNet allows using any publicly available pre-trained DenseNet models. The proposed architecture adds a LiDAR data stream to DenseNet that mirrors the RGB stream of the encoder until the fusion of LiDAR and RGB occurs. An additional layer is added for concatenation. The implementation allows joining the data either before the actual processing starts or before any of the 'DenseBlocks'. Smooth integration is achieved by first concatenating structurally equivalent preprocessed RGB and LiDAR data. Using 1x1 convolutions, the number of channels is subsequently halved, resulting in DenseNet's original number of channels. The implementation provides a method of investigating the influence of data fusion after processing each data type independently to a certain extent and then joining them at corresponding levels. As part of the decoding process, transposed convolutions and nearest neighbor upsampling are employed with a subsequent refinement to recreate the original size of the image, as explained in chapter 2. As an output of this model, class-wise heat maps are produced, equivalent to non-binary segmentation masks.

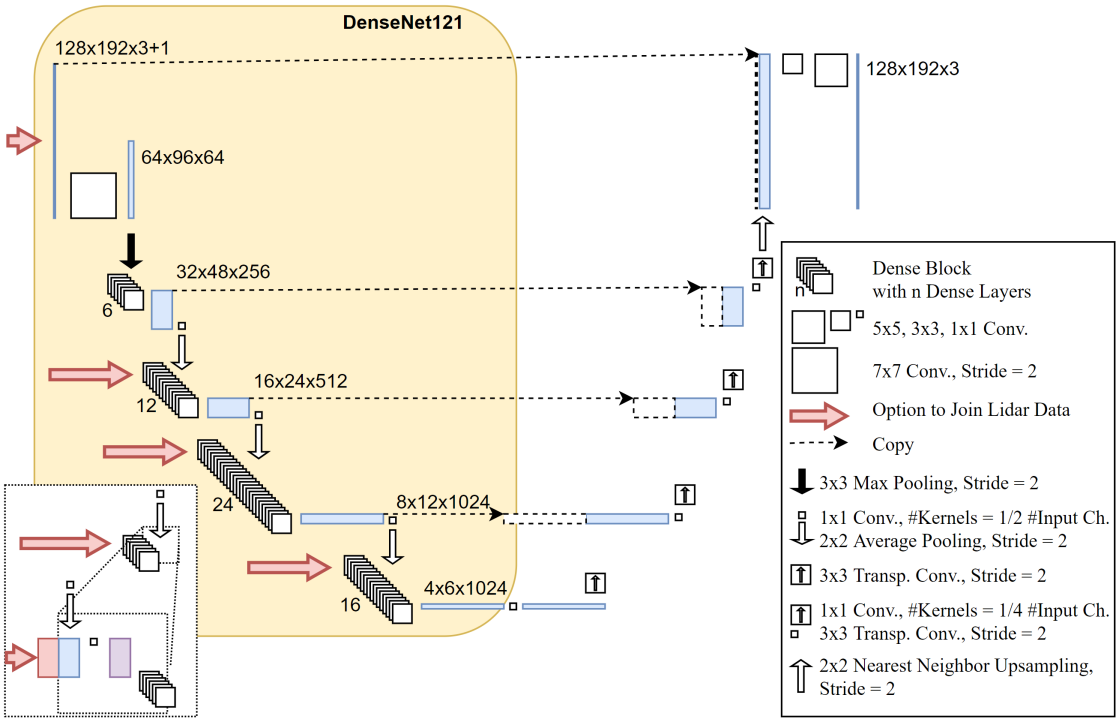


Figure 5.2: Network architecture: The encoder is based on the original implementation of DenseNet121, with an optional stream mirroring the encoder until the fusion site. At the fusion site (dotted box), LiDAR (red) and RGB (blue) data are concatenated, and 1x1 convolutions are used to halve the number of channels, resulting in a representation (purple), allowing to preserve of the network's original processing stream after joining both data types. Shortcut connections feed the data from the encoder directly into the corresponding site of the decoder (dashed boxes). Their additional abundance within the DenseBlocks makes the network highly compact and robust. If not denoted differently, parameters are set to the following values: #filters=1/2 #input channels, stride=1, padding=(filter size - 1)/2

Optimization

As a result of the architecture, an end-to-end trainable model is defined. To counter vanishing gradients, the weights are initialized using the Kaiming method [86]. Initialization of the original DenseNet component of the model is performed by using pre-trained weights provided by torchvision. A Sigmoid layer preceded by a pixel- and channel-wise binary cross-entropy was used to calculate the loss, resulting in class independence. The learning rate is fixed at 1e-3, as pre-trained weights of DenseNet are employed, and 1e-3 equals the final value, after learning rate decay, used by the authors of the original DenseNet [89]. The optimizer is Adam with standard values (beta1=0.9, beta2=0.999, eps=1e-8) [110]. A Google Cloud VM instance with an Nvidia Tesla K80 and 24GB VRAM is used for training. Focal loss is evaluated to address disparities in class occurrence and size because FocalLoss was designed to optimize for these kinds of imbalances

[111]. Hard positive mining is another method of addressing class imbalances [112], which is evaluated in addition to FocalLoss. Hard positive mining involves training on artificially selected subsets that contain more examples of a particular class than the training set does naturally.

5.4 Results, Evaluation, and Discussion

Comparing Early, Mid, and Late Fusion

The naming conventions of low, multi-level, and object-level sensor fusion used in chapter 2 refers to systems. In contrast, for fusion within a single network, the choice is to adopt the nomenclature of early, mid, and late fusion, as found in literature [96, 97, 98, 99, 100]. Early fusion is implemented as the simple concatenation of an RGB image and the corresponding LiDAR-based depth map. As the term implies, mid-fusion refers to joining both types of data near the middle of the encoding scheme. Late fusion is typically used to describe fusing two data streams directly before the final classification layers. In this present use case, late fusion refers to using a network configuration that joins the data at the final stage of the encoder. The network performance is evaluated using the Jaccard index, which measures the intersection over union (IoU) [107] of the ground truth mask and the predicted heat map. IoU values are averaged over all images with a union greater than zero pixels. In the following, the given orders correspond to the IoU values of the (early, mid, and late) configurations. As visualized in figure 5.3, there are vast discrepancies between the performance of pedestrians and cyclists, respectively, compared to vehicles. Averaged over all network configurations, IoU values for pedestrians (0.37) and cyclists (0.23) are significantly lower than the 0.85 value for vehicles. While vehicles (0.84, 0.86, 0.84) and cyclists (0.23, 0.22, 0.25) are detected approximately equally well in all network configurations, pedestrian scores decrease as a function of fusion depth: (0.41, 0.38, 0.33). In terms of performance relative to fusion depth, there may be a significant difference between pedestrians and vehicles, as well as between cyclists and pedestrians, due to the different mechanisms employed to create the ground truth masks. In the case of cyclists and vehicles, the bounding boxes are projected directly into the mask, while pedestrians are represented by a function that approximates the silhouette of a human.

Optimization Techniques

As a result of optimizing with FocalLoss instead of binary cross-entropy loss, all evaluated hyperparameter configurations resulted in unstable training. This hypothesis is supported by the estimates given in the RetinaNet paper, which indicate that the imbalances are too substantial for FocalLoss to sufficiently address them.

Attempts to leverage hard positive mining result in overfitting. This is explained by the following: Requiring only 1% of an image to contain the respective object classes to be considered for the positive mining subset only leaves 171 images of cyclists, equivalent to about 1,5% of the original training set.

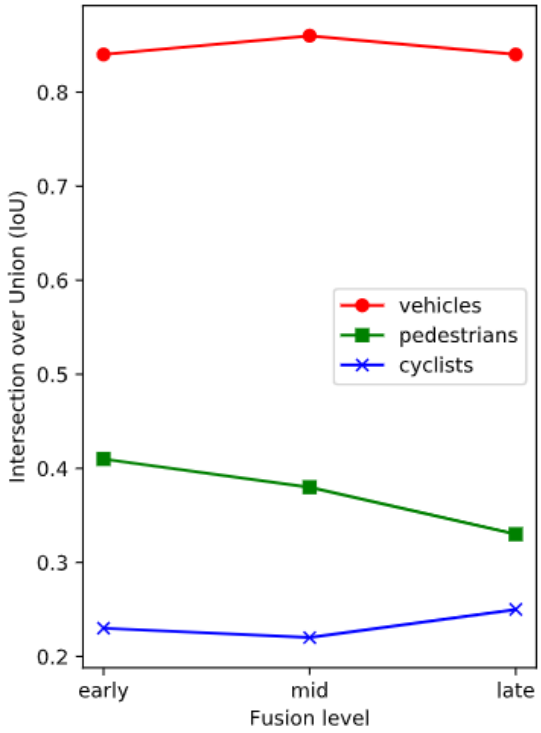


Figure 5.3: Prediction performance as measured by the IoU evaluated regarding the influence of the multi-modal fusion depth. Neither vehicle prediction nor cyclist prediction changed significantly by employing different fusion mechanisms. The earlier RGB and LiDAR data were fused in the network, the better the predictions of pedestrians. Overall performance is by far the best for vehicles.

Performance Evaluation

Due to the vast disparity in results between object classes, additional analysis is conducted to refine the performance assessment beyond the IoU level. In examining the performance differences of the network when considering the number of pixels belonging to a particular class, two aspects become evident:

- Firstly, different thresholds are set, requiring an image to contain a minimum number of pixels of a particular class, comparable to the approach of selecting data for a positive mining dataset. The abovementioned mid-fusion model is evaluated on these artificial subsamples of the validation set. As shown in figure 5.4, the class-wise IoU improves as a function of the minimally required pixels.

- Secondly, plotting the class-wise IoU for each image with respect to the number of pixels revealed that the model detects larger objects well. All pedestrians were successfully detected when applying a minimum coverage of at least 1% of pixels.

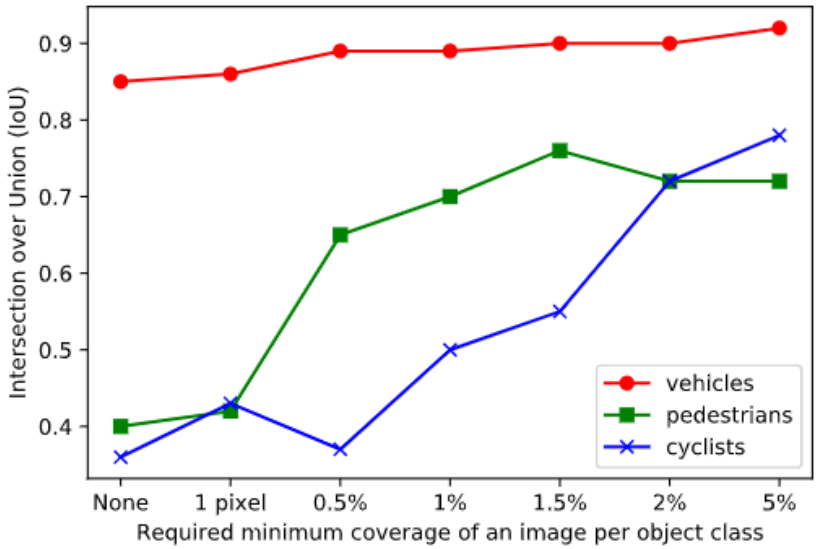


Figure 5.4: Prediction performance, as measured by the IoU, on thresholded subsets of the validation set. Each subset is only comprised of images containing at least the number of pixels required by the threshold. The performance improves significantly, requiring only 1% of each image to contain an object class.

The first observation is in line with other research. It has been established that one of the major weaknesses of one-stage object detectors is their inability to detect smaller objects. The second observation may be explained by the fact that the pure number of pixels, and consequently the generated loss, is larger for the biggest 5% of cyclists than for the entire bottom 95%. The same rationale applies to pedestrians.

By requiring only 1% of the image pixels to contain one or multiple objects of one class, the IoU performance is significantly improved. Performance increased as follows: for vehicles from 0.85 to 0.89; for pedestrians from 0.4 to 0.7; for cyclists from 0.36 to 0.5.

5.5 Conclusion and Outlook

Performance Discussion: Object Classes

It has been demonstrated in the results that the IoU significantly varies from one class to another. It can be seen from the dataset analysis that there are significant class imbalances, both in terms of the number of objects and their size. Despite the fact that pedestrians and cyclists are detected only mediocrely without any thresholds, their pixel-wise accuracy is highly accurate, exceeding the accuracy of vehicle maps. The lower performance in IoU may be attributed to an imbalance of background and foreground, which is especially pronounced for pedestrians and cyclists because bounding boxes only contain a subset of pixels belonging to the annotated object class. Consequently, simply

classifying a pixel in the bounding boxes not to be an object would result in higher accuracy values for the respective classes.

Combined with the performance improvements obtained by simply requiring an image to contain at least 1% of object pixels, this fact demonstrates the robustness of the network during optimization. Although the dataset presented adverse conditions, the network was able to learn a class representation. Adding to the adversities, employing binary cross-entropy as the loss function discounts smaller objects even further. The pixel-wise application of a loss function results in high loss values when the map contains one large object covering a large number of pixels instead of several smaller objects collectively covering a smaller number of pixels. As suggested by the authors of ResNet and DenseNet, the abundance of shortcut connections plays a significant role in the robustness of the model during optimization [84, 89]. Requiring at least 1% of image pixels to contain objects from one class results in a significantly improved IoU performance. As closer objects cover more pixels in the frame, making this assumption is not unreasonable in AD. This rationale is supported by the setup of the Waymo Open dataset challenge, categorizing bounding boxes into three difficulty levels. Concerning the threshold, it is important to note that a single object is not required to cover 1% of the pixels. It is the object class that needs to cover 1% of the pixels overall. Thus, the threshold is more reasonable, making the requirement more applicable to real-world scenarios.

Performance Discussion: Fusion Depth

The impact of fusion depth on evaluation results differed vastly for the different object classes. As shown in figure 5.3, only pedestrian detection benefitted significantly from changes in fusion levels. Performance on the respective maps was best for the early fusion architecture, joining the data before any separate processing of LiDAR and RGB data. The corresponding network for this early fusion setup has the least number of parameters of all configurations. Joining the data at later stages requires an additional LiDAR processing stream, resulting in doubling the parameters of the encoder.

Relating these findings to the differing functions generating the ground truth representations, it is argued that early fusion leveraging human domain knowledge on object appearance benefits the network in semantic segmentation tasks. The ground truth masks generated for vehicles and cyclists were simple one-to-one projections of the bounding boxes, whereas pedestrians were modeled in more detail using domain knowledge, better estimating a segmentation mask only covering the object.

The choice of more complex, exact silhouettes of objects, as found in semantic segmentation, requires a more fine-grained prediction compared to bounding box prediction. Bounding boxes cover background pixels in addition to the object, and projecting entire bounding boxes into segmentation masks results in more non-object pixels being wrongly annotated with the object class label. Consequently, evaluating object detection on a bounding box level, single-pixel differences have a very low impact on the overall performance. However, for semantic segmentation, single-pixel differences are important and easier for the network to detect when employing early fusion. By merging both sensor

modalities early on, the network is expected to be able to better associate information, simplifying the training process.

Conclusion

The findings on early fusion within a NN confirm that the approach of fusing data originating from different sensor modalities with a low level of pre-processing is suitable to preserve information. This leads to improved detection and segmentation performance, which confirms the results achieved by the LLSF as introduced in chapter 4.

Among the goals of evaluating this approach was the exploration of the use of object detection datasets for the training of semantic segmentation architectures. An architecture built for semantic segmentation was successfully trained on an object detection dataset based on bounding boxes. A key component of the proposed approach is the conversion of bounding boxes into heat maps leveraging domain knowledge. To close the gap between primitive bounding boxes and highly accurate segmentation masks, it is necessary to use more elaborate functions to make large-scale datasets like ImageNet accessible to the research community working on segmentation for pre-training.

Future Work

One approach to generating class-wise heat box functions would be to utilize the ground truth information from established semantic segmentation datasets. More specifically, the suggestion is to scale bounding boxes of each class to the same size and then average them over the segmentation masks contained within. This would result in heat-map-like approximations of an object class. A more elaborate attempt would additionally consider the bounding boxes' ratios, distinguishing, for example, cars as seen from the side from those imaged from the back. A second approach would be to first train a segmentation network using a semantic segmentation dataset. The segmentation head would be trained only on the provided bounding boxes, thus downscaling the problem to segmenting small image patches, i.e., identifying the object mask within the bounding boxes. The resulting segmentation predictions could be utilized as ground truth heat maps.

The main goal was to assess the impact the fusion level has on performance. The focus was not on comparability to object detection and semantic segmentation architectures. In future efforts, evaluating the approach on the Cityscape [108] and Kitti [91] datasets would be a priority for better comparison to other approaches. Due to the ideas incorporated in the network architecture, ideal candidates for reference architectures would be Faster RCNN [49] implementation with a DenseNet [89] backbone, FuseNet [101], and U-Net [90]. For a state-of-the-art comparison regarding semantic segmentation, Mask RCNN [50] would be a candidate. This architecture was designed for AD datasets, which contain a limited number of object classes. Evaluation of the architecture on datasets containing more object classes will likely require using a more powerful decoding module.

In the future, it would be interesting to see how much the additional certainty information each pixel contains, compared to bounding box detections, can be leveraged in advanced applications such as tracking.

6 Design of a Multi-Stage Perception System for Autonomous Driving

Highly assisted and autonomous vehicles rely on different sensor systems to observe the environment, with ML providing the semantic context of the detected objects. Over the last decade, deep learning has risen to prominence with the ability to map the complex, non-linear relationships between large volumes of observations and effectively classify these. However, given the bounds of efficiency and practicality in automotive applications, there remain several challenges for the effective deployment of object classification systems, not the least of which is the computational complexity. In the following, a solution for undertaking object classification in automotive environments is proposed, which is geared toward computational efficiency and effectiveness. The proposition uses a cascading hierarchy of traditional shallow ML classifiers, focusing on the use of expert domain knowledge both to build the classifiers and to motivate the correct selection of classifiers. This chapter outlines the motivation and scope of the problem, then introduces the proposition and outlines results obtained from simulation and the real world.

6.1 Introduction

The automotive environment has been developed around human drivers, whose biological sensing apparatus has evolved over generations to be able to understand the dynamic, cluttered, and partially obscured world. Furthermore, humans inherently adapt to the collective behavior and norms of their environment. Despite their high cost and advanced nature, even experimental sensors yield only a low-resolution, discretely sampled digital approximation of our dynamic world, significantly underperforming relative to human sensory and perception systems. Autonomous and highly assisted vehicles are expected to operate within this complex environment based on data captured by their sensor set, posing one of the main challenges to their realization. Most vehicles utilize a variety of sensors that observe the physical state of the vehicle and objects in the environment. Of particular interest for the use case of AD is the use of externally observing sensors such as camera, LiDAR, and RADAR systems, as explained and demonstrated in chapters 2, 4 and 5. Each of these sensors has disparate properties and capabilities from a spatial, spectral, and temporal perspective. The sensors are used to generate estimates of different physical properties of targets with a level of certainty. For example, RADAR systems offer good range and radial velocity detection characteristics but coarser angular detection. Conversely, LiDAR systems provide excellent range and azimuth detection but limited ability to measure the velocity of objects. Sensor fusion is generally used to combine the information generated by different sensor measurements. The process of

aligning sensors, associating detections, and then fusing the information allows a system to combine the knowledge provided by several observations in order to reduce the overall uncertainty. However, to make this knowledge useful in the context of a highly assisted or AD task, the semantic meaning of the object should be understood. That means that the vehicle sensors should not only measure the physical position and velocity of objects in the environment but also infer the type of those objects to determine how to plan the driving task accordingly. For example, consider a stationary vehicle on the emergency lane of a highway. This relatively innocuous occurrence may require only minor updates to the driving task. However, a pedestrian moving around that same vehicle may require a different set of actions, both in how the vehicle dedicates sensing and processing resources and in how the vehicle trajectory is planned. The concept of learning the semantic meaning of objects within the automotive environment is of substantial interest to the automotive industry [113]. It is the subject of extensive research in academia and industry. Specifically, applications of deep learning have become increasingly popular in recent years. Unfortunately, this comes with several drawbacks. These include immediate effects, such as the need for extensive processing capacity on the vehicle, and secondary effects, such as the cost necessary to collect and label the training data and the necessity for training systems with massive datasets. The following sections will elaborate on some of these challenges and set the context for the multi-stage classification concept.

6.2 Problem Statement

Highly assisted and autonomous vehicles work under the general framework of intelligent agents: they sense, interpret, and subsequently act within the environment. In the first case, sensors with disparate physical and spectral properties are used to detect objects. ML is used to assign semantic meaning to each object, hence classifying it. NNs, and in particular deep and convolutional NNs have been used in various applications in recent years. Such networks have demonstrated much potential in object classification. However, they have also introduced new challenges, especially within the context of a high-performance, low-power automotive-grade system. A NN is generally a simple system that maps the relationship between input features and output classes. This is achieved using a collection of connected processing nodes, referred to as 'neurons', where each neuron acts as a simple processing unit, operating with an activation function and bias. Neurons are organized into layers, and the connections between them are weighted, where the values are determined during the training process. The training process essentially exposes the system to data, where the output classes are labeled, allowing the training algorithm to determine effective weights for each connection in the system. The general aim is to produce a numerical system that models the effective relationship between the input data and output classifications. An introduction to NNs is given in chapter 2 and an in-depth description of the technical operation and the concepts of NNs is given in [114]. An introduction to Deep CNNs is also given in chapter 2, and [115] provides a more detailed introduction.

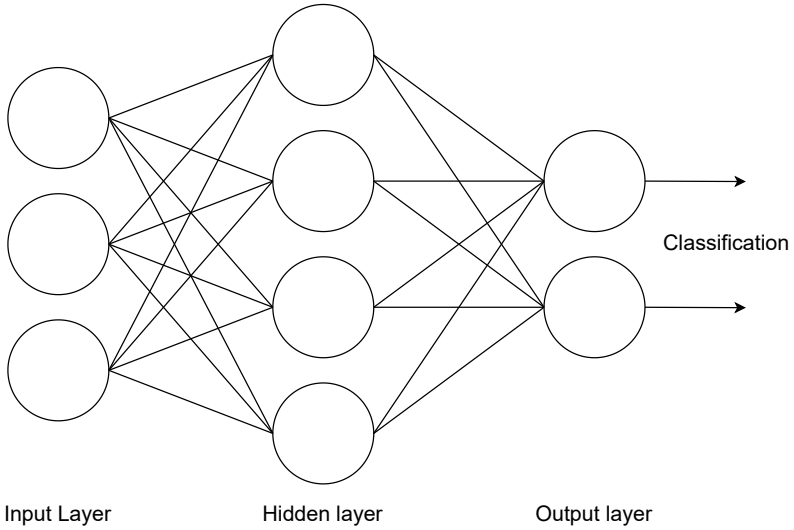


Figure 6.1: A simple single layer feed-forward neural network, with three input nodes, two output nodes/classes, and a single hidden layer with four nodes.

Classification problems range from very simple representations to very complex ones. When considering that width and height of an object are extracted from an input sensor signal, these two physical properties and the relationships between them might be used within a very simple NN to classify the likelihood of an object being a car or not. Cars will generally be observed from different angles in the automotive environment. The number of features required to effectively classify a car from various angles would have to be increased compared to a classification task with a fixed angle. This, in turn, requires a greater number of neurons in the hidden layers and potentially more hidden layers to effectively model the numerical relationship between the input features and the output classes. When looking at the common representation of a neural network as outlined in figure 6.1, where the input features relate to output classes through connections with a hidden layer, it is apparent that the numerical model represented by the neural network simply relates the input features to the output classes. For an artificial system to be considered intelligent [116], it must be able to generalize, correctly classifying different observations of the same class. Consequently, it should be able to generalize different examples of the same class, different measurement modalities, and different measurement conditions. For the concept of generalization to be realized, the first step is to record and label sufficient training data, including all modes of observations, different examples of the same class, and different noise characteristics. It is important to note that it is very difficult for NNs to learn characteristics that have not been observed as they have no basis from which to model the numerical relationship between input data, the derived features, and the output classification.

Convolutional layers provide a framework for transforming the input data into different features, which are often correlated and describe spatial relationships between different

input values. A mathematical theory describing the general convolutional transforms for feature extraction is outlined in [117], while [118] indicates that different observations of the same classes generally share correlated features. As a result of large volumes of training data and significant features describing each class, large numbers of neurons are organized into deep and wide neural networks, with multiple hidden layers and large numbers of neurons in each layer. This allows the effective and non-linear mapping of different features and the correlation between different features to output classes. While CNNs have excelled at a number of classification tasks, particularly in automotive scenarios, these applications have introduced new challenges. The most immediate of these is the vast computational resources necessary to build, train and utilize such networks [119]. The classification tasks enabling AD depend on computational power far outside the cost and power envelopes associated with designing and building vehicles at scale. Within academia, there is a general trend towards deeper and wider neural networks, with the supposition that a greater volume of processing neurons with the associated training data will allow greater capability in terms of inference across a greater number of classes within a wider variety of observation conditions. The 'BBD100K' dataset [120] consists of over 120,000,000 images organized into over 100,000 sequences. This, in general, requires a greater number of neurons to effectively map the non-linear relationship between input features and output classes and has resulted in a general trend for larger numbers of neurons in networks [121].

6.3 Pattern Matching

The realization of CNNs within AD functionality remains a substantial challenge. This is principally due to the large volumes of training data necessary to cover corner cases, which are likely to cause the system to misclassify, and due to the substantial computational expense associated with using CNNs in any capacity. The computations generally require substantial electrical power consumption, and it is very expensive to offer real-time capability for high-resolution imaging systems. In this context, real-time is considered the ability to fully process one information frame before the next frame is available. One of the underlying hypotheses for achieving generality in ML is that a greater volume of training data can be used to replace the required input for a system in terms of expert domain knowledge. This would mean that with sufficient data, the system can identify patterns in the input data and derive general numerical rules or transfer functions from those patterns. For a supervised learning problem, the system performs inference based on known and learned properties of the expected classes. A NN simply models the non-linear numerical relationship between input features and output classes. For deep learning systems with large numbers of different classes, the system learns the full relationship of features with all classes. Such a system returns, for each input example, a result for every possible class that is available to the system.

This chapter aims to demonstrate how an effective classification system that works computationally efficiently and has a low classification latency can be realized. The executed research indicates that the mapping between input features and inferred classes is

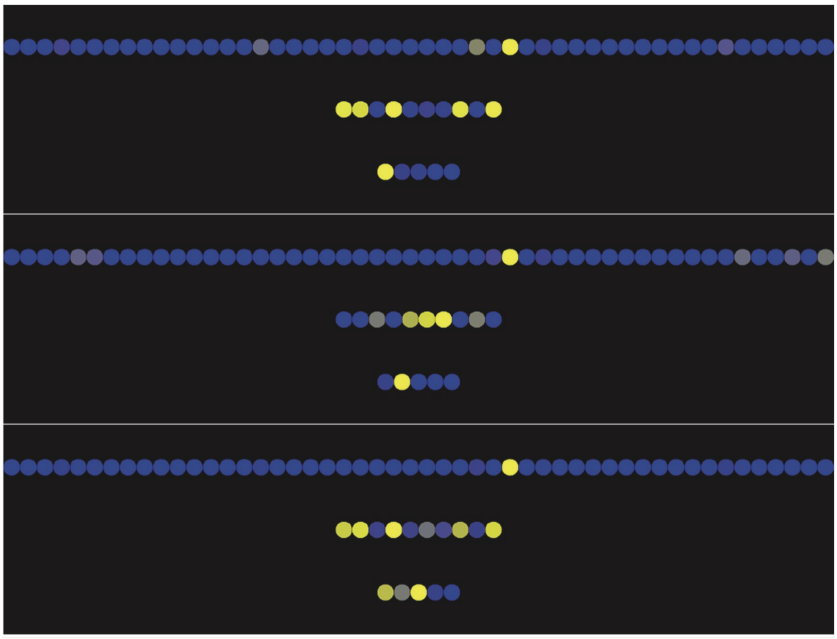


Figure 6.2: The neurons that fire for a typical classification task. This example shows three different classification paths for three different classes. Yellow neurons are active grey neurons that do not fire at all.

generally dominated by a relatively small number of neurons and firing paths. Furthermore, the neurons and pathways vary depending on different observable features. Figure 6.2 outlines an exemplary classification path within a real-world dataset. One can see that even though this is a relatively simple network, only a small portion is utilized for a given classification task. The aim is to introduce a classification strategy that is technically effective yet computationally efficient with regard to latency and computational costs. This is achieved by considering the following technical concepts: The first concept uses fused data from multiple sensor modalities to provide as much context to the system as possible. The second concept leverages expert-selected input features to provide both implicit and explicit labels, while the third concept applies a cascading architecture of specific classifiers. Using classifiers designed, trained, and subsequently tuned for a specific classification task is key to this concept, allowing a network to be smaller and shallower in its design and with an expectation of greater confidence in classification. Greater confidence is attained by avoiding the use of a pedestrian classifier that has been concurrently trained to detect and classify multiple irrelevant classes. The three main design concepts are described in more detail throughout this chapter.

Specific Classifiers

A guiding principle of the proposed classification approach is that NNs are a simple pattern-matching tool. The numerical relationship between input features and output

classes is optimized during the training process. During the inference of a new example, the network then predicts the likelihood of the example belonging to a particular class. Using a highly specific classifier greatly reduces the volume of training data necessary to effectively train a neural network [122]. Within a given driving scenario, a dynamic management system is responsible for selecting the specific classifiers for meeting the classification requirements of the driving scenario, dependent upon the sensor data available for that object. Engineering knowledge is leveraged during the design of the system instead of aiming for generality in classification. It is necessary to design and implement a number of different specific classifiers using expert knowledge regarding the driving scenario and the classification task for both the design and inference stages. As outlined in the next section, these smaller classifiers are organized into a cascading architecture, allowing the system to classify objects at different levels of fidelity as required. In summary, the concept of deep learning using large volumes of training data is supplemented with domain knowledge, preparing the dataset according to the classification requirements and explicitly defining the ML systems' expected output.

Multi-Sensor Data Fusion Input

Multi-sensor data fusion generally consists of aligning, associating, and integrating sensor data from disparate sources as outlined in chapter 4. By knowing that two disparate and conditionally independent sensor measurements are correlated, it is possible to combine them in a way that gains information about the system and, consequently, the driving situation. In the context of the proposed cascading classification methodology, using multimodal, fused data as input allows the training features to be richer in information from an information context perspective. Consider that the convolutional operation in CNNs is numerically intensive yet provides important information as to the location of the object within the image while simultaneously providing its relative size. Within the context of this system, an object is detected and tracked using a combination of LiDAR, RADAR, and camera data. Using fused data ensures that the input data used for classification is more meaningful from a physical perspective and more feature rich. For example, the system provides the range and size of an object explicitly, which reduces the uncertainty of the input and helps to provide improved confidence in classification. The underlying idea is that if the input data is less uncertain and more feature-rich, the confidence in classification is expected to be less uncertain.

Implicit Labelling for Specific Classifiers

One of the key concepts for the multi-stage classifier is the use of implicit information in the available data. Consider, for example, a car on the road. It is possible to classify the vehicle based on its visual properties only. However, its location and orientation are also important in classifying it as a vehicle, particularly within the driving scenario. When analyzing the impact of regions of an image on the image-based classification, it becomes obvious that specific parts of the image have a stronger impact on the classification than others. Assessments of existing sensor data and subsequent classifications indicate that

regions significantly contributing to the task of recognizing an object as a vehicle include areas where the wheels make contact with the ground and the recognition that the sky is positioned above the vehicle. This is a basic example that helps illustrate that the information implicitly associated with the vehicle is as important as the explicit features of the vehicle itself.

Implicit information on labeled object data is leveraged using two key principles:

In the first instance, the training data is selected specifically for the classifier. For example, a classifier detecting vehicles on a highway will principally detect vehicles from the rear; thus, this particular classifier is trained using only data on vehicles from the rear.

Secondly, the training data is selected to include the implicit information relevant to the scenario. Hence, the training data is selected to include implicitly important information about its surroundings across all sensor modalities. Ensuring that the classifiers are specific and limited to a particular classification task ensures that all of the data included in the training set is considered relevant to the underlying classification task. An example is the recognition of pedestrian crossings. In this case, the relevant implicit information might include the specific pattern of the crossing itself, the presence of specific signs or traffic signals, and even the likely presence of pedestrians in the near vicinity. The classifier does not only learn what a pedestrian crossing looks like but also the various scenarios in which it might encounter one. This enables the classifier to make more accurate predictions in real-world driving conditions, where context can be crucial.

6.4 Cascading Networks of Classifiers

The classification system proposed makes use of multi-modal fused sensor data. Instead of feeding the data into a DNN, which can be seen as a black box delivering an output based on the given input without knowing any details about the formation of the result, the proposed approach uses a network of cascaded classifiers. Each classifier within the classification hierarchy is specifically designed and optimized to solve a particular classification task. Being specific about the classification task allows shallow and slim network architectures for the single classifiers within the classification graph, as explained in chapter 2. Knowledge graphs are explicitly leveraged to determine the classification graph. The proposed perception system is not limited to classifying objects in the sense of assigning these objects to a class. Still, it can also detect the object pose and object state based on the multi-sensory input, which inherently captures this information. The ability to detect the pose and state of an object requires more details during the classification task but is not required for every object that is in the environment of the vehicle. Rather, only for those objects that directly impact the driving mission. These objects require a higher level of detail in their classification, which can be achieved by traveling further through the classification graph while discerning the pose and state of objects of lower interest is an unnecessary overhead and a waste of limited computing resources. In this case, the cascading classifier network allows exiting the classification graph as soon

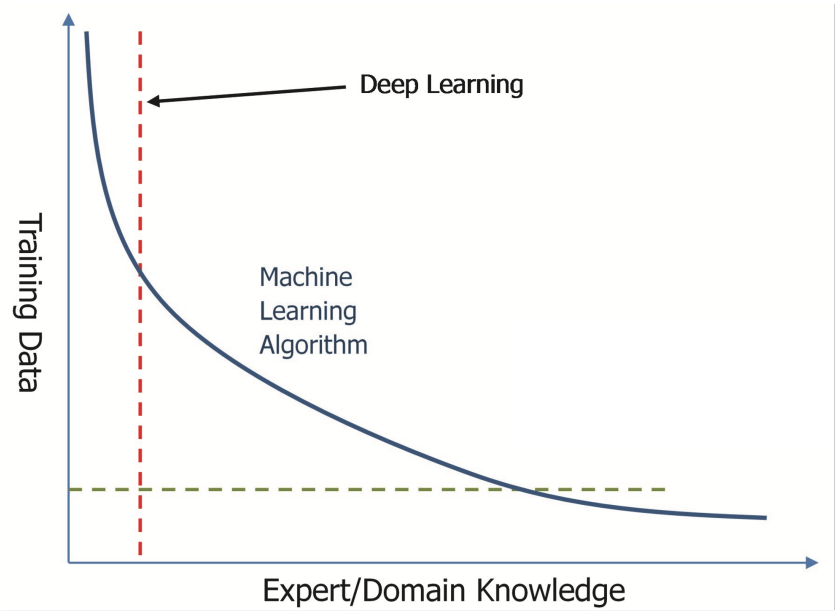


Figure 6.3: The relationship between training data and expert domain knowledge necessary for realizing an effective ML system.

as the necessary level of detail is ascertained for the driving application. Moreover, the cascading structure permits entry into the graph at specific levels when prior information about an object is available from preceding frames or other systems where the object has been identified and categorized.

Compared to DNNs, where generality is introduced by adding more neurons across more layers and increasing the amount of training data, the cascading network achieves generality by increasing the number of classifiers, depending on the classification task requirements. This reduces, on the one hand, the actual size of the neural network and the other hand, the amount of training data required to obtain satisfying results. The effectiveness of a ML algorithm is generally expected to be determined by the sum of the volume and variety of the training data and the amount of domain knowledge that has gone into the selection of training data and classifier design. Figure 6.3 outlines this relation, highlighting the specific classifiers selected in this example use case. To achieve the cascading network of classifiers, expert domain knowledge is necessary to analyze which features and which levels of details are required for a particular classification task. This knowledge is, in many cases, available or can be derived from existing datasets. An approach to integrating human domain knowledge can be found in [123]. The knowledge can be represented as a knowledge graph, as described in section 2.6, and the graph can directly be used to derive the individual classifiers of cascaded networks of classifiers.

A car on the road can be in multiple different poses that a DNN has to learn explicitly to be able to detect it. The cascading network architecture only introduces this level of detail about a car further down in the classification chain. It is difficult to detect corner cases, like a car that is upside down, with a DNN because common training sets do not

have any or very few examples of cars that are upside down. Using a cascaded network of classifiers, upside-down cars are not required in the training data. These cars will be detected as large stationary objects in the cascaded classifiers. The hypothesis is that it does not matter if a stationary object is a box or an upside-down car, the impact on the driving application is comparable for both types and in case these objects are impacting the driving path, a reaction can be triggered. The cascading design of the classification graph enables the perception system to be flexible and adaptable to the vehicle's environmental context and driving scenario. Flexibility here refers to the system's ability to be directed by regions of interest that determine the necessary detail level for object classification. Adaptability, on the other hand, signifies the system's capacity to modify its classification strategy and the classifiers used based on the prevailing situation. The combination of adaptability and flexibility, bolstered by the cascading network of classifiers, aims to enhance the efficiency and effectiveness of the perception system. This is achieved by improving the accuracy and confidence in classification outcomes while concurrently minimizing the computational and processing efforts required for essential classification tasks.

Returning to the vehicle example from the previous paragraph, consider the challenge of classifying the same vehicle, in the same physical space, during both day and night. While the objects are exactly the same, the sensory data recorded by a camera will be vastly different, necessitating different training data and classification methodologies. Instead of learning both cases within the same DNN, the cascading network of classifiers enables the perception system to use a specific shallow and slim neural network dedicated to classifying cars in well or poorly-lid situations.

Another advantage of the cascading networks of classifiers is reduced detection latency. As soon as a relevant object within the driving mission is classified as non-moving or safety-relevant, the information will be available to the successive components, such that the velocity can be reduced or an evasive maneuver is undertaken. More detailed classification information of the object is available with more specialized classifiers being executed. Still, the information most relevant to the driving mission, which might require a velocity reduction or steering correction, is available as soon as the classification at an earlier stage has been made. The cascaded classifiers offer the possibility to parallelize classification tasks at any desired level when necessary.

In addition, the cascaded network introduces possibilities for verification and validation of the perception system since the traversal within the classification graph is based on a set of rules making the classification results comprehensible. The fact that the cascading networks of classifiers use shallow and slim network architectures allows for tracking back the route of the classification results obtained by the cascade.

The benefits of the cascading network of classifiers are demonstrated using a school bus as an illustrative example, as depicted in figure 6.4. When driving on an urban road following a moving bus, detecting the school bus as a large vehicle is sufficient since it will behave comparably to every other vehicle within the driving scenario. When the bus reduces velocity, and the indicator appears, more knowledge about the object is required. Now the system needs to know that the large vehicle in front of us is actually a bus because traffic rules will change. With a changing driving scenario, the classification

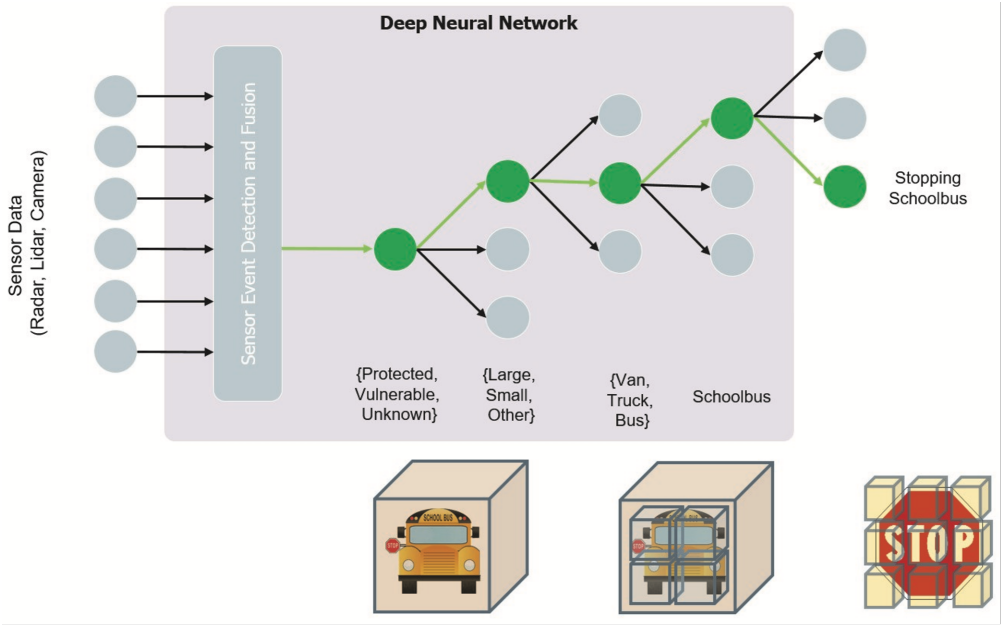


Figure 6.4: The concept of cascading networks of classifiers, outlining the idea that each specific classifiers (green circle) are a smaller section of the neurons included within the deep learning architecture. Each classifier is designed and trained for a specific training task using carefully selected training data.

has to be more detailed as the information that the large vehicle is a school bus is relevant in this situation. If the hazard lights are on or the stop sign is popped out in the United States, the vehicles behind are not allowed to pass the school bus, whereas passing the bus with reduced velocity is allowed in Germany.

However, within a normal driving scenario, classifying every bus to the level of detail where the hazard lights or the stop sign are relevant would require massive computing overhead and generally not be required in most cases. Summarizing the bus example, first, the object is identified as a large vehicle over several frames. When the vehicle begins to change its physical state by, for example, reducing velocity, the perception system then adapts to the new situation and identifies the large vehicle as a bus. This triggers the system to expend processing resources to better understand the semantic context of the 'large vehicle' and update its application accordingly.

6.5 Situation Adaptive Selection of Classifiers

Safely resolving driving situations by adequately classifying the objects in the surroundings is one of the key challenges of environmental perception. Especially in urban settings, the variety of possible objects that can be encountered and the absolute number of objects within the FOV of all sensors is enormous. Simultaneously trying to classify all objects to the same level of detail is computationally expensive. Depending on the

available sensor data, it is not always feasible due to computational constraints and limited signal quality.

Using the school bus example, in case the bus does not have the hazard lights flashing anymore, traffic rules change again. In this new situation, the region of interest needs to be adapted since the likelihood of people crossing the street, leaving, or trying to reach the bus has increased. The perception system's focus area for object classification must also be adjusted in response to this change. It now needs to identify individuals on the sides of the road and classify traffic participants in the passing lane.

Consequently, this work uses an approach to dynamically adapt a perception system to the driving situation based on the expected vehicle trajectory. The perception system uses three levels of granularity to classify objects in the environment. The level of granularity is determined based on three inputs:

- Vehicle speed
- Steering angle
- Expected steering angle based on the planned route and direction light

Figure 6.5 shows an example of a vehicle taking a right turn in an urban scenario.

The ROI extends with increasing velocity, which addresses the need to identify objects at greater distances at higher speeds. Computational constraints are less limiting in this context due to a sparser appearance of objects at high speeds, as, for example, found in driving scenarios on the Autobahn. In addition to the speed, the ROI expands based on the steering angle towards the steering direction. Especially in urban and rural scenarios, where larger steering angles are to be expected, this approach focuses the perception system on more critical areas. A vehicle turning at an intersection with pedestrians crossing is one such example. The steering angle is additionally adjusted based on the expected driving trajectory, as indicated by the route planning or the direction light. Especially in an urban setting, where vehicles turn after sitting at a traffic light, this occurs frequently and results in focusing on objects in more critical locations, like pedestrians crossing the street at a traffic light. An example of such an urban intersection and the respective ROI adjustment is shown in figure 6.5.

By establishing a minimum understanding of objects in the surroundings, scarce computational resources are used efficiently while ensuring a high level of understanding of objects relevant to resolving the driving situation. This relevance is approximated via the ROIs, and the approach can easily be extended with other relevance approximation measures, like event-based ones. Objects behind a vehicle moving away from the vehicle are less crucial than objects which will cross or approach the driving trajectory. The school bus example demonstrates this example well because a school bus behind the vehicle does not change traffic rules and can be treated like any other vehicle, whereas a bus ahead of the vehicle needs to be classified in more detail to resolve the driving situation and adhere to traffic rules as imposed by such a vehicle.

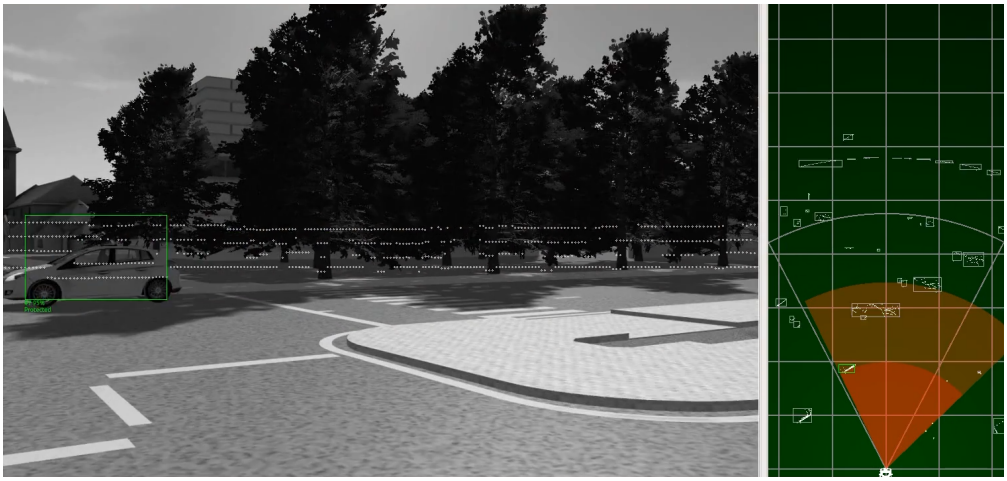


Figure 6.5: Situation adaptive ROIs

6.6 Testing the Framework

The conceptual framework of the perception system and the cascading networks of classifiers is assessed through the use of both simulated and real-world scenarios. These scenarios serve as testing grounds for evaluating the efficacy of the situation-adaptive cascaded classifier approach. The real-world scenarios cover driving situations on urban roads, including crowded crosswalks, intersections, and urban highways. Across these scenarios, the perception system is able to classify objects to the desired level of detail, according to their location in the vehicle environment, on a laptop PC in real-time. The results were produced using a basic implementation of the framework as shown in figure 6.6.

Test Scenarios

Real World Test Environment

Real-world driving scenarios were captured in various surroundings at different times of day and in changing environmental conditions, ranging from busy urban crossing to fog on the Autobahn, as shown in figures 6.9 and 6.10. This results in a dataset with various lighting and weather conditions. The data recording was performed by a vehicle equipped with LiDAR, camera, and RADAR sensors, which is depicted in 6.7. The mounting positions of all devices are known, and the sensor setup has been calibrated in advance. The extrinsic and intrinsic calibration matrices are known. The recording vehicle uses the following setup for data-capturing purposes:

- **Ibeo Scala LiDAR sensors:** In total, three Ibeo Lux LiDAR sensors are mounted front facing, resulting in a 180 degree coverage of the environment in front of the vehicle. The sensor returns frames at a frequency of $25Hz$.

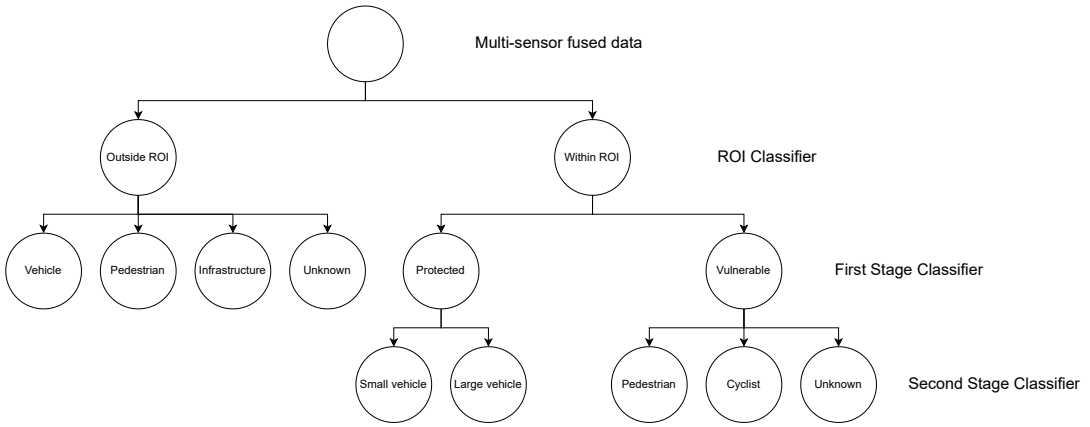


Figure 6.6: A simple and representative implementation of a cascading architecture for the multi-stage classification. This architecture was used to generate the results in this chapter. The black circles represent classifiers at different levels of detail.

- **Front smartmico short range RADAR:** There are two front mounted short range RADARs operating at a frequency of $24GHz$ with a FOV of 100 degrees. The sensor returns targets at a frequency of $30Hz$.
- **Front smartmico long range RADAR:** One front mounted short range RADAR operates at a frequency of $77GHz$ with a FOV of 100 degrees. The sensor returns targets at a frequency of $30Hz$.
- **Front FirstSensor camera:** The used camera is a grayscale camera with a resolution of 1280x800 at $30Hz$. The FOV is 55 degrees.

A perception system using multiple levels of granularity for classification, as achieved by implementing cascaded classifiers, is evaluated based on the ability to handle harsh environmental conditions while ensuring a minimum level of object classification capabilities. In the current implementation, a first-stage classifier provides a high-level classification of surrounding objects while using fewer classes to differentiate between objects. In the case of challenging environmental conditions, sensors are more prone to noise. Consequently, the classification quality decreases and the usable sensor data is sparser. A coarse determination of object classes has a higher likelihood of being successful in cases when sparse sensor data is available when compared to a fine-grained classification of objects due to the lower complexity of the transfer function required for the classification process.

Simulated Test Environment

The simulated test scenarios were created as closely as possible to the real-world scenarios and are based on simulated sensor data. The simulation leverages the Simcenter Prescan software, which provides a physics-based simulation platform [124]. It is used across the

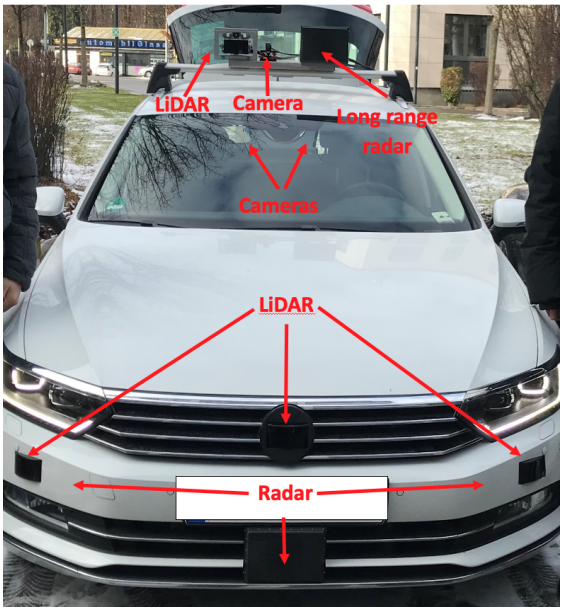


Figure 6.7: Test vehicle setup

industry to develop, test and validate ADAS and AD systems. In contrast to real-world circumstances, the scenarios generated with Simcenter Prescan are quantifiable, controllable, repeatable, and environmental conditions can be adapted. Examples are shown in figures 6.5 and 6.8.

- **Camera:** The camera data is emulated following a physics-based approach to sensor simulation. Simulating camera data entails the replication of the physical device through modeling elements like the lens, color filter arrays, image sensors, circuit board, and associated noise models. The underlying simulation platform is Unreal Engine [125]. Objects have physically correct properties to take ambient light models, geometries, and the definition of materials and how they interact in a visible wavelength into account.

- **LiDAR:** The physical device is simulated by modeling the beam cross-section, the emission pattern, the scan pattern, and the pulse shape of the used hardware. The simulation engine is based on a ray tracer based on path tracing algorithms. Objects are similar to the camera simulation and use material models that describe the behavior of light interaction at different visible and non-visible wavelengths (855nm, 905nm, 1550nm).

- **RADAR:** To simulate a physical device, the modeling includes the antenna pattern, the location of transmitters and receivers, the specified waveform, and the operating frequency. The simulation engine uses a ray tracer similar to what is used for the LiDAR simulation. Objects have geometrical specifications, and ma-

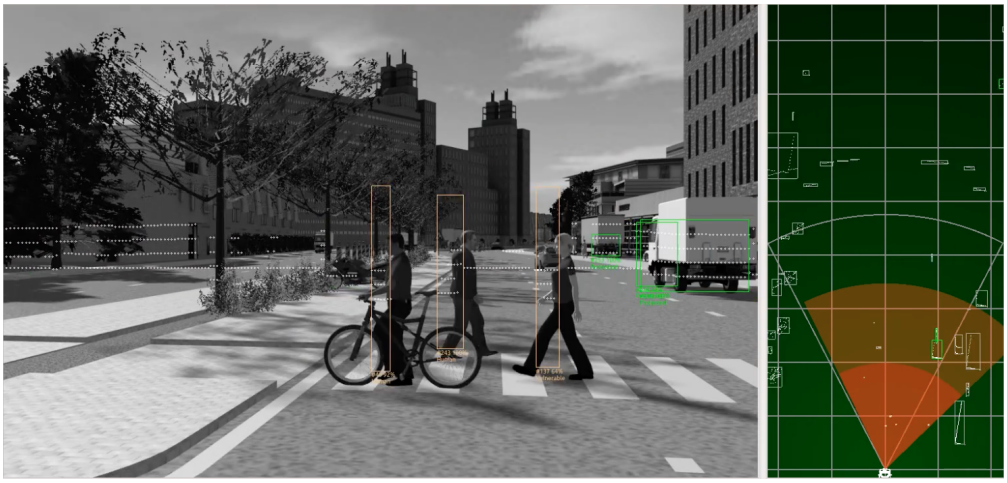


Figure 6.8: Simulated urban pedestrian crossing

terial property descriptions include the proper electromagnetic permittivity and permeability to ensure that RADAR wave propagation is simulated accurately.

Test Results

To test the concept of a multi-stage classifier, fused sensor data from multiple sources is used, as described in detail in chapter 4. Following the architecture outlined in figure 6.6, the logic to classify objects is described below:

- Fused objects are first classified as to whether they are 'objects of interest' or not. Essentially this is a level 0 classification that does not consider the objects themselves but their physical location relative to the vehicle.

- Objects outside the region of interest are classified using a simple LiDAR based classification scheme derived from [126]. Objects are classified as either a vehicle, infrastructure, or pedestrians.

- The cascaded classifiers are used to classify objects within the region of interest (ROI), whose maximum size is confined to the camera's field of view. Objects within the ROI deemed less critical are classified by the first-stage classifier. Conversely, objects within the ROI considered most critical are exclusively classified by the second-stage classifier. The classifications include small/large vehicles, pedestrians, cyclists, and the category of unknown.

The test results of the cascaded classification system for the real world and the simulated data are shown in table 6.1.

Table 6.1: Results for cascaded classification

	Crosswalk		Intersection		Highway at day		Highway at night with fog	
Cascaded classifier level	1	2	1	2	1	2	1	2
Real world detection rate in percent	99.3	99.1	98.7	97.9	99.8	99.3	98.9	96.7
Simulation detection rate in percent	99.7	99.6	99.1	98.6	100	99.9	100	99.5

Qualitative Discussion of Results

Qualitative results of the multi-stage classifier are shown in figure 6.9 and 6.10. The scenario shown in figure 6.9 is an urban crossing scenario observed from a stationary vehicle. It demonstrates the performance of the perception system in an urban environment where different levels of detail are required depending on the location of an object relative to the vehicle. The first stage classifier identifies vulnerable and protected road users in the less critical ROI (orange). The second stage used for the most critical ROI (red) classifies objects as small/large vehicles, pedestrians, cyclists, and unknown.

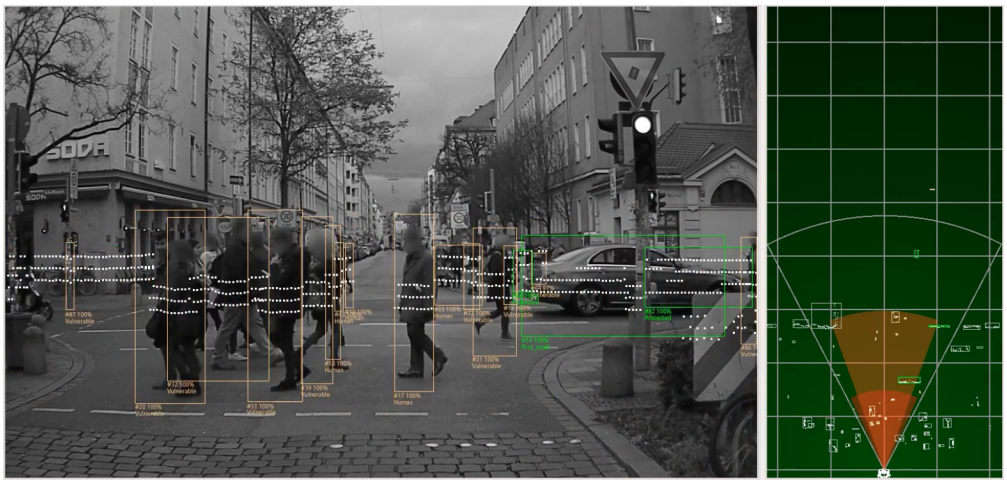


Figure 6.9: Urban Pedestrian Crossing: The left side depicts the objects as seen from the driver's perspective, while the right side presents the objects and ROIs. In this context, the critical ROIs are represented in red, and the less critical ones are shown in orange. The right-side representation offers a top-down view using the projected LiDAR data of each object.

The example in figure 6.10 demonstrates a coarse-level classification with sparse sensor data. The scene was recorded on the Autobahn at night while experiencing foggy conditions. This poses a challenge to both a human driver and a perception system.

The first stage classifier of the perception system is able to identify both vehicles as protected road users leveraging LiDAR and RADAR based features for classification

only. Image data is not used in the first stage classifier. Operating the vehicle based on the classified sensor data is possible in such a scenario because the basic object classes required to plan the vehicle trajectory are available.

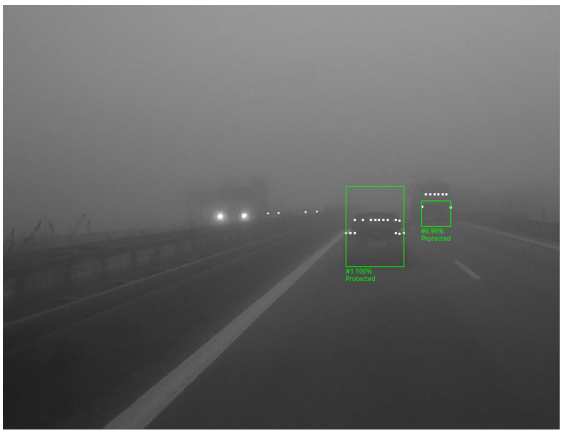


Figure 6.10: Fog on the Autobahn at night

Figure 6.8 shows an urban intersection based on simulated sensor data. Similar to the real-world example, the level of classification granularity depends on the criticality of the ROI with respect to the driving scenario. Additionally, figure 6.11 shows an example with a high incidence of road safety posts. Those pose the danger of being misclassified as pedestrians using a simple classifier, like the first stage. In literature, similar misclassification issues are also observed of pedestrians being classified as trees or poles [127]. The results, as illustrated by the red boxes in the fused LiDAR data on the left, highlight this aspect because of wrongly classifying the safety posts. The results of subsequent stage classifiers, depicted on the right in figure 6.11, employ the cascading network along with a more intricate input derived from LiDAR, RADAR, and image data. In this instance, the perception system unambiguously categorizes them as unknowns. In the simulated camera image, all bounding boxes for the posts are delineated in white, signifying their accurate classification as unknown.

The qualitative findings underscore the advantages of employing cascading networks of classifiers, substantiating their effectiveness and efficiency in classification tasks. Objects initially classified with sufficient confidence may no longer need to be classified unless the sensor data, driving scenario, or driving application changes. For example, the object may be tracked and move from outside the region of interest into the region of interest, or additional sensor data might become available.

6.7 Conclusion and Outlook

Over the last decade, the availability of larger datasets for training and increased computational capacity has impacted ML applications. The trend has resulted in utilizing deeper and wider networks, which aim to move towards greater generality. However, for

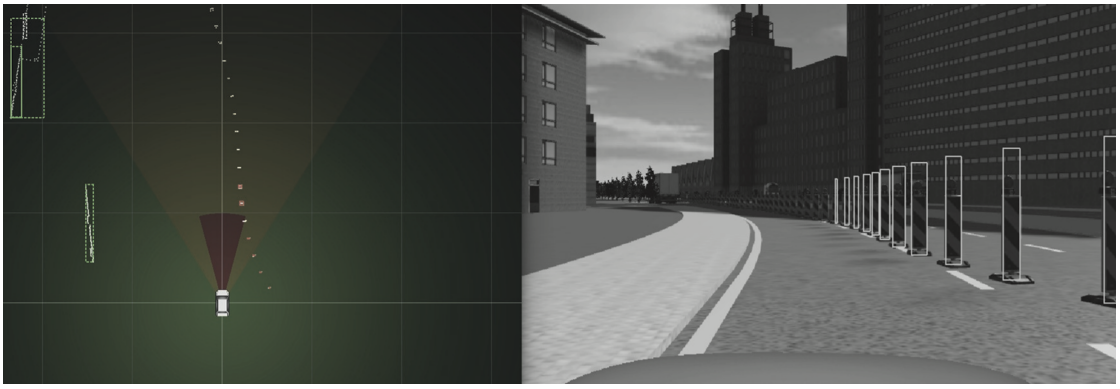


Figure 6.11: Results of the multi-stage classification system using simulated data

such systems to be truly useful in real-world applications, such as AD, they must be effective, reliable, and practicable.

The proposed concept of the multi-stage cascading architecture treats the classification problem as a simple pattern-matching one, split into multiple stages addressing different levels of classification granularity. The proposed cascaded system is specifically designed by human experts and includes the following:

- Cascading hierarchy
- Classification granularity requirements
- Individual classification problems
- Training data
- Features within the training data

Consequently, the presented approach replaces the generality expectation with expert domain knowledge. The results indicate that the multi-stage classifier is a suitable architecture for classifying objects of interest in the automotive environment. The results show that the perception approach works under various environmental situations in urban and rural settings while experiencing varying weather conditions. This has been demonstrated based on simulated and real-world data. This approach has value for AD applications like in-vehicle or stationary perception systems. However, the value of the proposed approach is not limited to one domain. Future work is expected to show the benefits of leveraging domain knowledge in a cascaded classification approach for additional applications and across industries.

The results indicate qualitatively that the proposed methodology reduces the computational overhead for in-vehicle systems. This is the case due to the reduced volume of data to be processed. While it might be possible to use traditional academic metrics, choosing metrics specific to the automotive domain is expected to provide a more qualitative assessment for future work. In addition to more detailed performance assessment

work, further research is required to expand the concept and extend the logic necessary to transition across the classification hierarchy for different driving applications and adapt to the varying availability of sensor data.

7 Pedestrian Detection System for Racing

The primary focus of this chapter is the integration of the components proposed throughout this thesis, demonstrating their combined functionality and tangible value in real-world applications. The AD technology discussed throughout this thesis is further subjected to the rigorous and demanding environment of rally racing. The successful performance of the technology under such conditions would underscore its robustness and versatility.

7.1 Introduction

The sport of rally racing is popular worldwide and fascinates millions of people regularly. Thousands of spectators attend each race, which is spread out over many rally stages, some of which are more than 25 kilometers in length. Due to the nature of rallying, a wide range of environmental diversity is found, and passages are secluded from the driver. In such areas, many fans attempt to take pictures of the cars and get dangerously close to them. According to the Federation Internationale de l'Automobile (FIA), half of all motorsport fatalities in the past decade are related to rally disciplines, with the majority of fatalities being spectators [128]. Clearly, there is a need for pedestrian detection in dangerous areas so that rally event organizers can take appropriate action and improve the overall safety of rallies.

To accomplish this goal, it is essential to implement technical solutions to detect endangered fans. However, the harsh conditions encountered in rally racing present several technical challenges that need to be addressed and prevent the use of default AD components in rally racing. This chapter leverages and integrates approaches presented in chapters 4, 5, and 6.

7.2 Problem Statement - Sensing Challenges in Rally Racing

Rally racing presents several challenges for environmental perception, including sensor fusion, due to the variety of environmental conditions and erratic movements. An additional challenge is associated with rallying because rally fans hide actively to avoid being detected. The combination of these three factors creates one of the most challenging sensing tasks related to pedestrian detection. The conditions found in racing are only comparable to corner cases of regular street traffic.

The problem of varying certainty and uncertainty in sensor measurements is a common problem in safety-critical applications such as those found in regular vehicles and the same applies to the system described throughout this chapter for detecting spectators

during rallies. Consequently, it is essential to understand how well each of the used sensors performs under which environmental conditions. This understanding is crucial to design a reliable sensor fusion which is the basis for object detection and perception. Because of the high number of possible combinations, collecting this data from real-life driving scenarios would either be extremely expensive or even not feasible since some of the combinations are extremely rare. This challenge can be overcome by simulating the same scenario for various desired environmental conditions and sensor setups.

Challenges Introduced by Harsh Environmental Conditions

Racing takes place around the world in a variety of climates. The temperature may be as low as minus 30 degrees in Scandinavia or as high as 40 degrees in Middle or South America. It is, therefore, possible to encounter a wide variety of weather conditions, including snow, rain, fog, and bright sunlight. Rally stages are not only conducted during the day but also at night. As a result, numerous dissimilar environmental conditions are encountered, which poses a challenge for the perception stack. The condition of the roads is also an important factor to consider. The cars often race off-road and are exposed to dirt and gravel roads. Consequently, heavy and very sudden shocks are experienced at a high frequency, presenting considerable challenges when associating data from multiple sensors.

Challenges for Sensor Fusion and Object Detection

Sensors on the vehicle operate at different frequencies in an asynchronous setting. This temporal offset results in spatial offsets comparable to regular on-road vehicles. It is described in chapters 1 and 2. A properly executed sensor calibration process is necessary and a prerequisite for enabling sensor data association. However, in contrast to on-road vehicles, the sensors on a rally vehicle are exposed to more extreme stress caused by the track conditions in combination with high speeds and acceleration.

Rally vehicles drift around corners and frequently hit holes, tree roots, and comparable obstacles on such a track. Additionally, some rally stages include jumps and corrugated roads, which cause torsional forces on the vehicle, directly resulting in offsets in the sensor positions compared to the state of calibration when the car was not in motion. A small offset in sensor position results in a much larger offset in the to-be-associated sensor data. This has various implications for the underlying sensor fusion approaches and their robustness, as described in chapters 2 and 4.

In addition to the extrinsically induced uncertainty regarding spatial offsets, the wide range of environmental conditions introduces uncertainty in the reliability of the sensor measurements. Erroneous sensor measurements can negatively impact all following processing steps in the pipeline if not handled properly as described in [129]. Due to the possibility of adapting to environmental conditions, a sensor set containing a variety of sensor modalities is advantageous in this context. In cases when the vision is impaired, the likelihood of receiving faulty or low-quality data from the sensors is increased. A similar problem may also occur in regular cars due to snow, rain, fog, and sun glare.

In racing, dust is an additional factor to be taken into account. Some tracks have sand or dirt surfaces, and high vehicle speeds cause dust clouds, impairing the sight and, therefore, the vision-based sensors of following vehicles. It is important to note that the vision system is considerably more error-prone in such cases, impacting the association of sensor data.

Sensor data association and object detection are performed by a fusion network comparable to the one described in section 2.2. Based on the specific environmental conditions and the available sensor set, the association approach and fusion network described in chapter 4 is adjusted based on the insights gained from evaluating the performance of individual sensors under different conditions. Using low-quality sensor data, or relying too little or too much on one specific sensor modality, can lead to missed object detections, false positives, and multiple objects being identified as one. This has implications for all subsequent processing steps.

In both street traffic and racing, it is important to be able to adapt the sensor set and thus manage the uncertainty in the sensor data. Suddenly appearing targets in the planned driving trajectory of a vehicle are critical and need to be detected with a high level of certainty. An AD system would trigger emergency braking or change the trajectory to avoid a collision. In the case of following or oncoming vehicles, this potentially leads to dangerous situations. In addition to being distracted, a race driver who is wrongly alerted is at risk of making a driving mistake. A well-studied example to demonstrate the strength of an adaptive sensor fusion is identifying ghost targets for automotive RADARs as described in [33]. Manhole covers often cause false detections of objects in the RADAR domain. A sensor fusion system that relies too heavily on RADAR information may result in false positive detections of objects. On the other hand, employing a sensor set comprised of multiple modalities and sensors capable of utilizing all accessible information can enhance confidence in affirming the absence of objects along the driving trajectory.

Challenges for Perception

A key challenge for AD systems and racing is the reliable perception of objects in the surrounding environment to resolve situations safely. In both cases, imperfect and faulty sensor measurements complicate the classification of objects in the surrounding. It has been described that RADAR sensors are prone to falsely detecting objects. Environmental perception is, however, primarily concerned with classifying detected objects correctly. To meet the requirements for the desired applications, state-of-the-art NNs, such as YOLO or Faster-CNN [46, 49], are not able to classify objects adequately.

As far as well-lit driving scenarios are concerned, state-of-the-art classification approaches are effective. Nevertheless, under conditions of significant lighting variations or challenging weather, a variety of complications can arise. The use of vision-based perception approaches can lead to several false detections due to visual artifacts. A common error is the classification of billboards or street signs as pedestrians or spectators. The presence of wet surfaces and their resulting reflections, puddles, and vegetation are other sources of false positives. Optical similarity to pedestrians can lead to a faulty classifi-

cation in all of these cases. It is difficult to debug and identify the underlying learned features that are causing this behavior in the first place since they are not necessarily human-perceivable. As seen in various ML applications, this is a common problem [130]. A richer feature set based on different sensor modalities can be used to achieve a more robust and reliable perception approach.

Hardware Limitations

A camera is one of the most popular sensors used in AD systems, as well as other systems aiming to detect objects and perceive the environment. However, as a vision-based sensor, a camera creates an image by passively letting light onto the images using a light-sensitive sensor. As a passive sensor, a camera is directly affected by poor lighting conditions. Due to the working principle of a camera, the image-capturing process is impacted by motion, which results in blurry images. Not only the absence of light but also too much light, as is the case when sun glare occurs, results in poor image quality.

There are two popular camera technology choices: rolling shutter and global shutter. There is a high likelihood of suffering from rolling shutter artifacts when a rolling shutter technology is utilized, negatively impacting object detection and perception. This demonstrates that the camera is an error-prone sensing technology. Especially when the camera is subjected to challenging weather conditions coupled with strongly varying lighting conditions, it becomes less reliable than in well-lit conditions. Additionally, the choice between grayscale and color cameras has a significant impact on performance. The capture frequency of grayscale cameras is higher than that of color cameras, and similarly, the dynamic range of grayscale cameras is greater than that of color cameras. Higher frame rates reduce blur and motion artifacts, whereas a higher dynamic range is advantageous for challenging lighting conditions. In some cases, the information provided by a color camera might be essential to safely resolve a driving situation. One of the disadvantages of the camera is that it is not possible to directly measure either the distance or the velocity of an object.

A RADAR sensor, in contrast, is directly able to measure the range, direction, and velocity by actively emitting and receiving electromagnetic waves while making use of the Doppler effect. The range is measured by the time delay of the frequency response of the reflection and is limited by the signal bandwidth. The measured directional resolution is limited by the antenna size, and the resolution of the radial velocity is limited by the measurement time. Electromagnetic waves are advantageous because they are capable of propagating through rain, snow, dust, and fog. Therefore, RADAR can provide accurate measurements under various weather and lighting conditions. Since electromagnetic waves are actively emitted, RADAR is considered an active sensor. The result is that it is less affected by some corner cases occurring in racing that could reduce the performance of a visual system. Heavy shocks, vibrations, or the like do not have a significant impact on the sensing capabilities of the system. Additionally, there are no motion artifacts, such as motion blur, as you would encounter in a camera image. A unique selling point of RADARs is that the sensor can be placed behind a bumper,

which means that a direct line of sight is not required. This is a significant advantage over cameras or LiDARs when placing the sensors on vehicles.

A change from 24GHz to 79GHz has improved the resolution of current automotive RADARs, a trend observed throughout the industry. As a result, more detailed information about the surroundings is provided. RADARs, however, have an angular resolution of approximately 5 to 10 degrees, making them much cruder than other sensors, such as LiDAR. A detailed analysis of automotive grade RADARs is provided by [131]. A further advantage of RADAR is that the detection range is greater than that of LiDARs or cameras. The performance of a system is particularly affected by this characteristic at high speeds. For example, if a distant object is detected, the information from the RADAR sensor can be used to focus the attention of other sensors in a certain direction for further analysis. In the case of an assisted driving system on the highway, the information provided by the RADAR might be sufficient to safely resolve the driving situation.

7.3 Digital Twin - Virtual 3D Environment

Digital Twin of a Race Track

The target is to create a digital twin of the environment for the purpose of simulating and evaluating a wide variety of scenarios while leveraging the approaches introduced in chapters 4 and 6. The process of building the digital twin involves two major steps:

- As a first step, the environment is scanned with LiDAR, imaging technology, and differential GPS. The surroundings are captured from the ground level and from a top-down perspective using a drone. In addition, this information is combined with coordinates from the differential GPS and the point cloud derived from the LiDAR data. It is important to capture not only the road itself but also the surrounding area to be able to simulate realistic scenarios. There were no road users on the track while recording the data. Consequently, the first step results in a colored point cloud of the road and its surroundings.

- In the second step, a digital twin of the track is generated based on the data captured in the first step of this process. The model is an open-scene graph model [132], and it is ready for use in simulations. Based on the resulting model, evaluation scenarios are developed. An example of the result is shown in figure 7.1

Build Scenarios

The digital twin enables the creation of a large variety of different scenarios. The possibilities range from slightly changing the surroundings to simulating various weather conditions.

For the present use case, a total of 32 sections make up the stage of the rally course. As part of the evaluation process, scenarios are selected to contain a wide variety of turns,

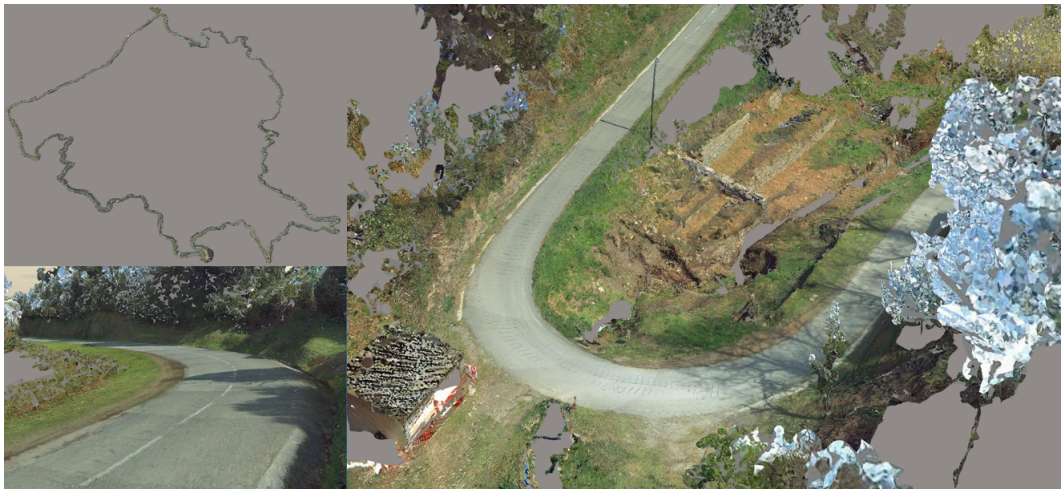


Figure 7.1: Digital twin of the rally stage

junctions, forks, and bends in the road. During the process of building the scenarios, road users, such as cars, trucks, bikes, and pedestrians, are included in respective scenarios. Additionally, it is possible to alter the vegetation slightly by adding trees or shrubs that were not present in the original model. One scenario included the addition of a shrub, which blocked the direct line of sight from the camera to a pedestrian.

There are 17 sections where spectators are present during a rally event, which are used as simulation scenarios. The spectators in a scenario are simulated as adults standing or walking. Each scenario is grouped into one of the following risk groups:

- Danger: At least one pedestrian is on the road or within one meter of the road
- Warning: At least one pedestrian is between one and three meters away from the road
- Info: At least one pedestrian is more than three meters away from the road

Each scenario is simulated 18 times, which is equal to the number of variations evaluated for each scenario with regard to pedestrian risk and environmental conditions. In four cases, however, spectators cannot be placed more than three meters from the track, which results in twelve variations for those scenarios. Each scenario is based on real-world observations to ensure the appropriate placement of pedestrians.

Based on the digital twin model of the course, the scenarios were created as closely as possible to real-world scenarios. The simulation is based on Simcenter Prescan software, which provides a physics-based simulation environment. A more detailed explanation of the simulation engine and its properties is given in chapter 6.

In the simulation, sensors are located in any desired location on the vehicle or the track as part of roadside units. Vehicle dynamics and spectator movements are part of the simulation. An example of a scenario simulated from different perspectives is shown in figure 7.2

(a) In-vehicle view

(b) Simulated in-vehicle view

(c) Simulation top-view

(d) Simulation overview

Figure 7.2: Digital twin

Sensor and System Configuration

Following the creation of the scenarios, it is necessary to define the sensor configuration. Sensors are placed in two different ways. The first option is to mount the sensors in a stationary road site unit. The other option is to use a Computer-Aided Design (CAD) model of the vehicle to enable an onboard simulation. A stationary sensor system does not change its location over time; only the objects in the model move relative to the sensor. Nevertheless, the value of simulating a stationary system setup is to better understand how the sensors should be placed on a roadside unit. The FOV and the exact mounting positions are determined.

To enable a direct comparison of the evaluation results, the simulation uses the following sensors for the stationary and in-vehicle systems, where all in-vehicle sensors are mounted front-facing:

- **Velodyne Puck Lite** The LiDAR sensors is front-facing, resulting in a 180 degree coverage of the environment in front of the vehicle. The sensor runs at a frequency of $20Hz$ with a horizontal angular resolution of 0.4 degrees. The vertical FOV is 30 degrees with an angular resolution of 2 degrees. The sensor has 16 layers and operates at temperatures between -10 and 60 degrees Celsius.
- **Delphi ESR 76.5 GHz RADAR:** The RADAR is operating at a frequency of $76.5GHz$ with a horizontal FOV of 90 in short range mode and 20 degrees in long range mode. The sensor runs at a frequency of $20Hz$ with an angular resolution

of 1 degree vertically and 2 degrees horizontally and in short-range mode with a range accuracy of $1m$ and 5 percent. The vertical FOV is 4.2 degrees in short-range mode and 4.5 degrees in long-range mode.

- **Front smartmico long range RADAR:** One front mounted short range RADAR operates at a frequency of $77GHz$ with a FOV of 100 degrees. The sensor returns targets at a frequency of $30Hz$.
- **Front camera with generic imager:** The used camera is a grayscale camera with a resolution of 1920x1080 at $30Hz$. The horizontal FOV is 120 degrees, and the vertical FOV is 60 degrees.

For this use case, a Delphi ESR sensor was selected as a RADAR sensor because it is able to switch between long and short-range modes, while it is not possible to run both modes in parallel.

Stationary System

The sensors are mounted on a pole that is $400cm$ high. Since the scenarios are based on real-world data with varying topology, the sensor poles are placed in different locations with varying height differences from the road surface. Consequently, the pitch of the sensors varies and is adjusted so that the scenarios can be captured as desired. Due to the underlying assumption that the sensor pole can be positioned in an ideal manner, yaw and roll are not considered. As a result, the vector representing the pole is orthogonal to the assumed ground plane. Figure 7.2 shows the stationary unit from different perspectives, and 7.3 illustrates the covered area per sensor and where spectators move during the simulation of the scene.

Table 7.1: Mounting height of the sensors for the stationary system

	Camera	LiDAR	RADAR
Mounting height in mm	3600	3800	4000

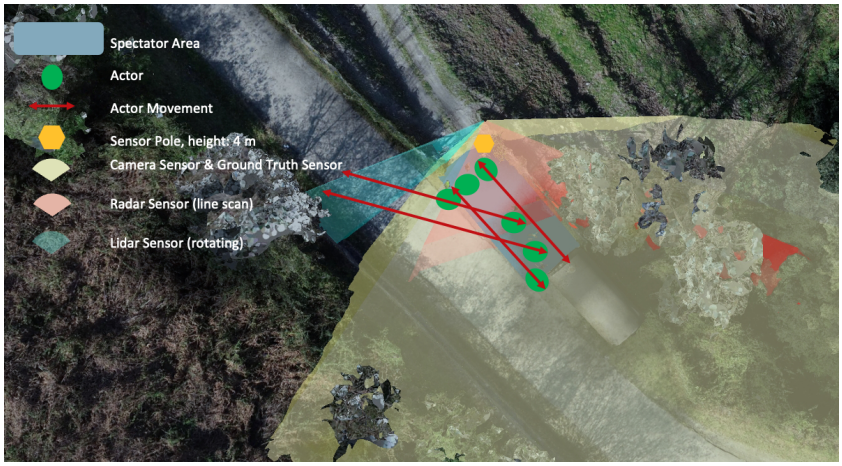


Figure 7.3: Simulated environmental from a birds-eye perspective

In-Vehicle System

The mounting positions of the vehicle sensors are shown in table 7.2. Coordinates are relative to the center of the vehicle coordinate system, which is the middle of the rear axle at ground level, following best practices as found in the industry. Sensors are positioned so that the driver's field of vision is not impacted in any way. The sensors are mounted higher up to minimize the impact of environmental conditions on them.

Table 7.2: Scenarios

	Camera	LiDAR	RADAR
Mounting position x-axes in mm	1800	1959	1950
Mounting position y-axes in mm	938	900	978
Mounting position z-axes in mm	1290	1388	1395
Pitch in degrees	-3	-1.8	-0.5

Set Environmental Conditions for Simulation

The simulation engine provides the possibility to change the environmental conditions while the scenarios and objects within the scene remain the same. This means that dynamic objects follow the same trajectory. Each scenario is simulated and evaluated during the day and night. The following weather conditions were selected for daylight conditions:

- Clear sky
- Fog
- Moderate rain

- Heavy rain

For nighttime scenarios, clear skies and heavy rain are simulated and evaluated. The simulation engine simulates the light beams emitted by vehicle lights and the resulting lighting conditions. Figure 7.4 shows examples from within the in-vehicle perspective.

The in-vehicle system is evaluated for all 17 sections where spectators are present, simulating the above-mentioned weather conditions during day and night for three risk categories for spectators. This results in a total of 282 simulation runs, as four scenarios can only consider two instead of three risk profiles for spectators. All 32 sections are part of a rally stage, and when the simulation was run, the remaining 15 scenarios were also included. Because of the complexity of the simulation, a cloud computing environment was used to execute it.

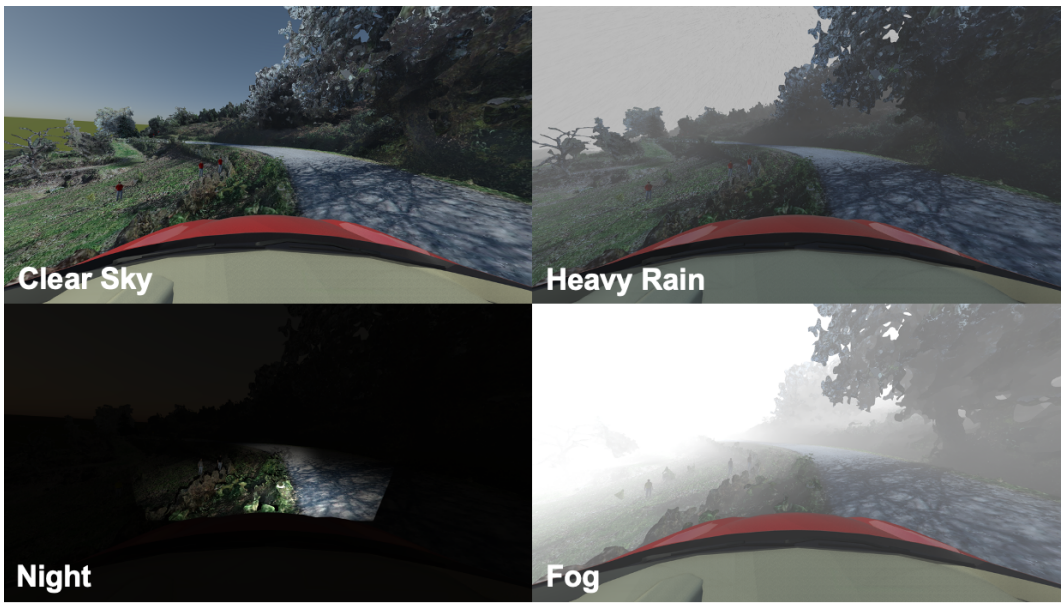


Figure 7.4: Simulated environmental conditions from an in-vehicle perspective

7.4 Sensor Fusion and Perception Approach

Sensor Event Detection and Fusion

The simulation provides data from RADAR, LiDAR, and camera sensors, which are provided at different frequencies. The low-level sensor fusion approach, as described in chapter 4, is leveraged to fuse the information from these different sensors. The fusion network, however, is adjusted based on the available simulated data. LiDAR and RADAR branches of the Bayesian network found in chapter 4 are similar, while the camera branch has undergone substantial changes. The Bayesian fusion network uses the likelihood of refined bounding boxes containing an object. A region proposal

network (RPN) leveraging anchors serves as the basis for providing object hypothesis. This approach is introduced and explained in [49]. An anchor is centered at the sliding window, and the proposed default setup with nine anchors per sliding window position is used. Anchors are ranked according to the likelihood of containing an object. An RPN produces regions, which are classified into foreground and background. Additionally, the bounding box regression coefficients refine the bounding box itself, and non-maximum suppression reduces the redundancy of bounding boxes.

The presented approach uses a Faster RCNN network which is trained end-to-end on the COCO 2017 dataset using a ResNet-50 backbone. The bounding boxes with their respective detection scores are used as the leaf node of the camera branch of the Bayesian fusion network. Consequently, the second layer of the Bayesian network is a leaf node, and the network architecture is simplified compared to the network introduced in chapter 4. Nevertheless, using the Faster R-CNN addresses the previously identified limitations of the low-level sensor fusion approach as explained in chapters 4 and 6.

Multi-Stage Classifier for Spectator Detection

The classifier used for this purpose is depicted in figure 7.5 and utilizes the framework of the multi-stage classifier described in chapter 6. By default, the sensor setup covers the relevant areas, so the classifier does not distinguish between regions of interest. There are two stages in the multi-stage classifier. Objects are classified as vulnerable, protected, or unknown in the first stage. Because protected and unknown objects are not relevant to the current use case, they will not be classified further. The protected road users include pedestrians, cyclists, and others. For the present application, the pedestrian is the most important class since pedestrians make up the vast majority of spectators. There is, however, a possibility of encountering cyclists or other vulnerable groups, such as people on scooters.

A multi-stage approach is chosen due to reduced computational complexity and a reduced number of false positives, as explained in chapter 6. Especially warnings based on false positives would negatively impact a driver and could lead to distraction and driving mistakes.

7.5 Evaluation of Stationary and In-Vehicle Systems

The stationary and in-vehicle systems leverage the same approach for sensor fusion and perception, working in concert to create an overarching system for spectator detection. Although both systems share the same foundational approach, their operation differs in two primary aspects: the sensor placements and the impact of vehicle movement on the in-vehicle system.

To more thoroughly comprehend the performance and limitations of each system, they are evaluated individually. This evaluation utilizes a detection rate that hinges on the successful detection and classification of a spectator, as explained in the previous section.

The integration of these two systems gives rise to a unified system dedicated to detecting spectators in potentially hazardous areas. Consequently, this evaluation method

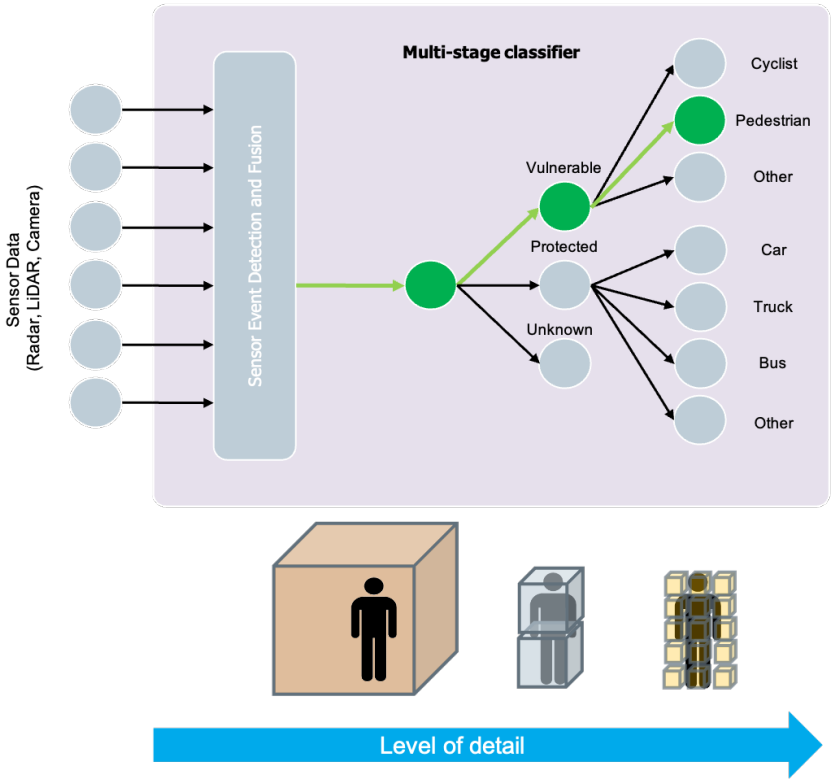


Figure 7.5: Multi-stage classifier for spectators: Example of pedestrian detection

has been chosen. Thus, the true positive detection rate emerges as the predominant and relevant performance indicator in this evaluation context. The basis for evaluation is the simulated sensor data using the digital twin of the track. The detection rate for the respective system is given, as well as the detection rate for the camera-only system for reference.

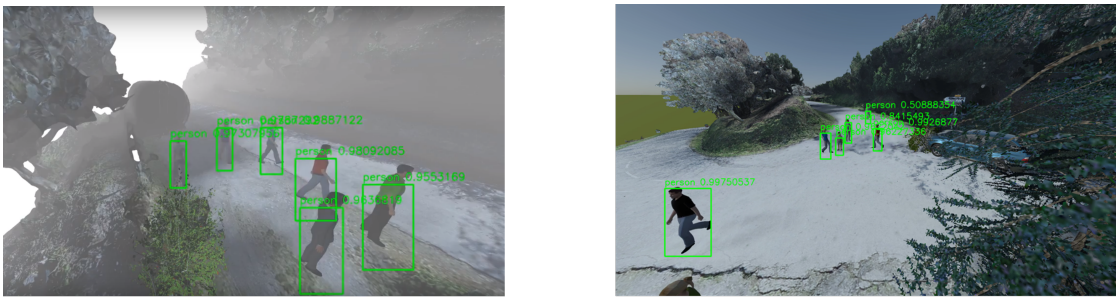
Stationary System Evaluation

For the evaluation of a stationary system, various environmental conditions are taken into account. Compared to a camera-only approach leveraging a Faster RCNN, the multi-modal sensor system provides higher detection rates. The results are shown in table 7.3 and are frame-based. There is no tracking of detections across frames.

A 98.7% detection rate is achieved by spectator detection in daylight conditions, and performance deteriorates as weather conditions worsen. In rainy conditions, the overall performance remains at 86,6%, although the detection rate drops as the intensity of the rain increases. During foggy conditions, the performance of the system is comparable to that of light rain. The camera-only approach performs best in daylight, while rainfall intensity decreases the detection rate. The performance of light rain and fog is comparable, with the performance of foggy conditions being lower. Regarding nighttime, the

Table 7.3: Results for stationary system

Environmental conditions	Conditions	System detection rate	Camera detection rate
Daylight/Daytime	**Overall**	**98,7**	**86,3**
Rain	**Overall**	**86,6**	**50,2**
	Light rain	89,5	58,6
	Heavy rain	81,8	36,2
Fog	**Overall**	**87,1**	**51,5**
Night	**Overall**	**93,4**	**9,8**
	No rain	96,9	14,9
	Moderate rain	87,6	1,2



(a) Section 21 with fog

(b) Section 26 at daylight

Figure 7.6: Stationary results

multi-modal system performs significantly better than the camera-only system. Based on the combination of sensors, an average detection rate of 93,4% is achieved at night, while the camera-based approach results in only a 9,5% percent detection rate. Both approaches are adversely affected by rain at night. A simulation including fog and an example for a sunny day are shown in figure 7.6. All pedestrians present in both scenarios are detected.

In-Vehicle System Evaluation

The in-vehicle system is evaluated based on environmental conditions and FOV impairments caused by the conditions and objects within the environment. Compared to the stationary system, occlusions caused by objects in the surrounding are an important aspect to consider during evaluation. A camera-only based approach using a Faster RCNN architecture serves as a reference when evaluating the in-vehicle system. The results are based on a per-frame detection without tracking detections across frames and are shown in table 7.4.

The in-vehicle system performs slightly worse for all environmental conditions when compared to the stationary system. This is expected due to the setup of the sensors

Table 7.4: Results for in-vehicle system

Environmental conditions	Field of view impairment	System detection rate	Camera detection rate
Daylight/Daytime	**Overall**	**94,6**	**67,5**
	Clear line of sight	98,7	96,8
	Partial occlusions (grass/bushes)	86,3	43,4
	Sun glare	93,9	4,2
	Poor contrast or separation (spectator close to objects/ blending in with the environment)	91,2	32,8
Rain (light to heavy)	**Overall**	**76,4**	**42,2**
	Clear line of sight	76,8	56,8
	Partial occlusions (grass/bushes)	73,7	19,3
	Poor contrast or separation (spectator close to objects/ blending in with the environment)	79,5	15,0
Fog	**Overall**	**82,4**	**46,5**
	Clear line of sight	84,6	63,1
	Partial occlusions (grass/bushes)	76,6	24,9
	Poor contrast or separation (spectator close to objects/ blending in with the environment)	82,1	6,8
Night	**Overall**	**93,2**	**8,4**
	Clear line of sight	96,9	13,4
	Partial occlusions (grass/bushes)	85,2	0,0
	Poor contrast or separation (spectator close to objects/ blending in with the environment)	90,3	0,0

on the vehicle. For daylight conditions, 94,6% of pedestrians are correctly detected by the multi-modal system. During rain, the overall detection rate reduces to 76,4%, while foggy conditions reduce it to 82,4%. At night, the overall system performance is 93,2%, a slight decrease compared to daylight scenarios. At nighttime, the actively emitted light of the vehicle lamps is considered as a light source during simulation.

In cases where there is a clear line of sight, the detection rate is higher and at 98,7%. The detection rate drops to 86,3% in case obstacles in the surroundings impair the line of sight for one or more sensors. Whenever sun glare occurs while having a clear line of sight between sensors and objects, the detection rate is decreased compared to no glare scenarios. The poor contrast or separation category refers to examples where pedestrians are either too close to an object or when spectators blend in with the environment. Pedestrians being too close to other objects, like the metal tank in figure 7.6, affects the sensor signals and, therefore, the sensor fusion and perception. Blending in with the environment refers to cases where spectators wear dark clothes while the background appears in a similar color, which results in poor contrast. Consequently, the evaluation shows that poor contrast or separation reduces the system detection rate to 91,2%.

A 98.7% detection rate is achieved by spectator detection in daylight conditions, and performance deteriorates as weather conditions worsen. In rainy conditions, the overall performance remains at 86,6%, although the detection rate drops as the intensity of the rain increases. During foggy conditions, the performance of the system is comparable to that of light rain. The camera-only approach performs best in daylight, while rainfall intensity decreases the detection rate. The performance under light rain and fog is comparable, with the performance of foggy conditions being lower. Regarding nighttime, the

(a) Day **(b)** Night

Figure 7.7: In-vehicle results

multi-modal system performs significantly better than the camera-only system. Based on the combination of sensors, an average detection rate of 93,4% is achieved at night, while the camera-based approach results in only a 9,5% percent detection rate. Both approaches are adversely affected by rain at night.

Examples of the in-vehicle system evaluation for a daylight and a nighttime scenario are shown in figure 7.7. The daytime scenario shows spectators standing in shady areas next to the road. The nighttime scenario shows two pedestrians, one of which is not lit by the car lights and is hardly visible in the camera image.

Qualitative Result Interpretation

The results of the simulation are based on a rally stage with a variety of changing environmental conditions. In general, the simulation results demonstrate good performance under all conditions, but the performance for corner cases varies considerably depending on the scenario. It must be noted, however, that the simulation engine is not capable of handling certain corner cases in the real world. Extreme sun glare, combined with snow and icy roads, is beyond the simulator's capability. The direct comparison with real-world recordings illustrates that additional work will be required to evaluate how the overall system performs in such situations.

Several corner cases can be used to demonstrate the strengths and weaknesses of the proposed approach. Early object detection is one of the main strengths of this multi-modal and multi-stage spectator detection and classification system. Figure 7.8 shows a scenario when a pedestrian suddenly enters the field of view of the car from behind an obstacle blocking the view, clearly demonstrating the major advantage of the low latency required for object detection and classification. This behavior can be achieved under various environmental conditions, including darkness, thanks to the use of multiple sensor modalities. Pedestrians can already be detected by LiDAR or RADAR before being visible in the camera image. It is possible to observe similar effects at varying distances from spectators and under a variety of different environmental conditions. Examples are when vegetation partially or completely obscures the camera's FOV or when spectators are detected first using RADAR and LiDAR data at night.

(a) Top view and vehicle view before entering

(b) Top view and vehicle view after entering

Figure 7.8: Hidden spectator entering the track

The examples of spectators at night, as depicted in figure 7.9, show the value of using a multi-modal approach. The spectators are hardly visible in the camera image, while the RADAR and LiDAR are able to properly detect and classify spectators in this scenario. There is a scenario with a metal tank, as shown in figure 7.6, and it demonstrates some of the limitations of the proposed approach. A spectator who walks and remains right in front of the metal tank is not detected by the multi-modal sensor setup in case of impaired image quality. The disruption caused by metal reflection on the RADAR signal, originally intended to bounce off a pedestrian, in conjunction with the poor image quality, is insufficient for accurate spectator detection in such scenarios.

Limitations of the Simulation

Both stationary and non-stationary systems produce comparable results, contrary to what would be expected in real-world situations. This can be explained by effects, which are not addressed by the simulation engine.

Camera

Camera sensors produce results based on simulated data, as expected in real-world situations. The algorithm used to detect pedestrians could be improved by using a larger set of data from simulation and real-world scenarios. The implemented classifier used for the evaluation is trained on real-world and simulated data. However, real-world data dominates the dataset. This is a conscious choice because purely using simulated data would result in strongly overfitting the network to simulated data while increasing the detection level to a non-feasible level. The camera-based spectator detection works as expected, and the quality decreases with worsening lighting conditions. Future research should consider improving the ground truth data used for the simulated camera sensor to enable an improved evaluation. This could be achieved by implementing the following changes:

- Indicate whether an object is partially occluded for a sensor by adding metadata
- Introduce an occlusion ratio for objects

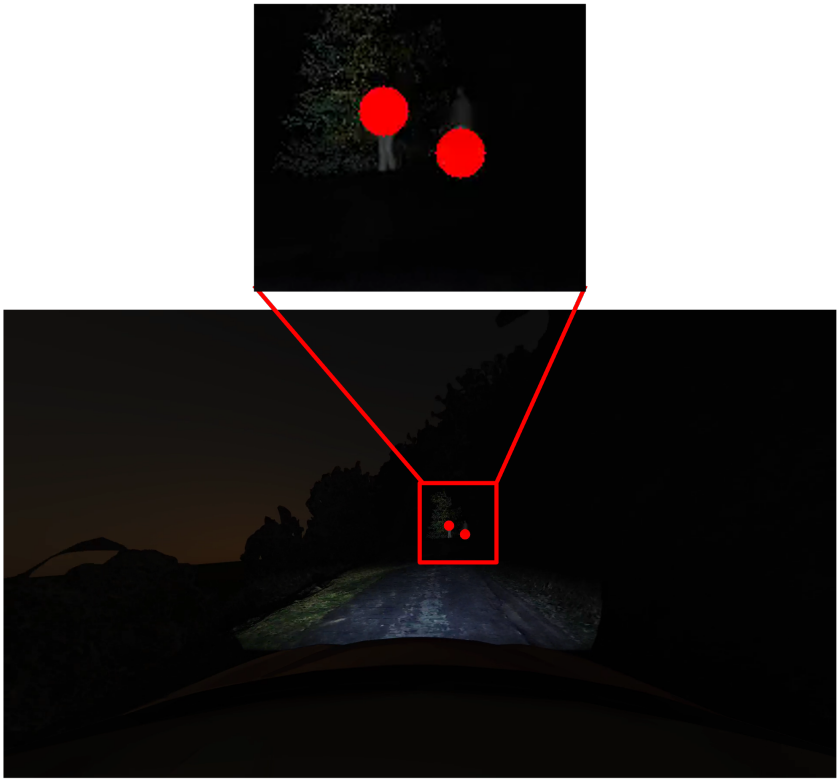


Figure 7.9: Spectators during nighttime: The red circles illustrate the RADAR detections.

- Include camera depth information

Corner cases can be better evaluated with an understanding of the occlusion of an object. Especially when objects are occluded in a subset of the mounted sensors, this information is beneficial to understand the overall system performance and limitations.

There is a strong correlation between the size of an object in the camera and the distance from the spectator. Therefore, the size of an object plays an important role in the ability to detect it. It is therefore recommended that depth information for the camera ground truth sensor should be included directly. As a result, it will be possible to develop a metric that illustrates how the detection rate of the camera depends on the distance from the object and, accordingly, the size of the object.

RADAR

Unlike real RADAR sensors, the simulated RADAR sensor lacks true-to-life SNR measurements, crucial for further processing and probabilistic sensor fusion. It is important to note that the simulation fails to take into account some real-world effects, such as multi-surface reflections of the electromagnetic wave or undesired attenuation. The present simulated data has a significant issue due to the absence of RADAR clutter.

There is a tendency for RADAR sensors to suffer from noise, which results in the appearance of ghost targets. To counteract this effect, RADAR targets and their properties are usually tracked over time. In general, one observation of the simulated data in comparison to real-world data is the ratio of pedestrian targets to the number of overall targets is significantly higher. Based on this, more targets from the surroundings would need to be included in the simulation, and the effects of undesired reflections of electromagnetic waves should be considered to achieve a more realistic comparison.

LiDAR

The LiDAR-based pedestrian object detection is strongly correlated with the classification capabilities based on camera data. The quality of LiDAR detection and classification directly depends on the distance of an object and if it is partially occluded. There are more LiDAR reflections available for objects that are closer to the sensor. To classify an object, the number of reflections is important. In general, the more reflections there are, the better the classification will be. Despite this, objects can be detected with as few as three reflections. There is a strong correlation between the number of reflections and the certainty of an object's presence. This is a crucial aspect of a probabilistic sensor fusion, as explained in more detail in chapter 4. A decrease in reflection density with increasing distance is accounted for by the implementation of the DBSCAN algorithm. The density-based clustering algorithm was sufficient for detecting all of the objects in the simulation for the present use case. Despite the simulation of challenging environmental conditions, such as rain, the quality of the LiDAR sensor did not degrade significantly as expected based on real-world observations. The jitter applied to introduce noise to the simulation does not correspond to a physics-based behavior. Furthermore, the reflection properties of the surrounding environment and the road were not taken into consideration.

7.6 Conclusion and Outlook

Assessment of both the stationary and in-vehicle systems under diverse environmental conditions underscores the effectiveness of a low-level sensor fusion approach paired with perception methodology, leveraging domain expertise for detection and classification tasks. Each of these systems clearly outperforms camera-only based detection and perception systems, particularly evident under demanding environmental conditions. Each system individually, as well as both systems in combination, enable unprecedented performance for spectator detection.

Each system on its own has applications beyond spectator detection in racing events. The in-vehicle system is relevant for AD applications requiring an object detection and perception stack. The detection of vulnerable road users is of high interest and crucial for the safe operation of automated vehicles. The stationary system has applications in urban settings, where roadside units detect and classify objects in the surrounding. These units communicate with traffic control centers and vehicles. This example shows the importance of combining stationary and in-vehicle systems for building a low-latency

digital twin of the environment, which is a crucial element in highly automated vehicles. This chapter demonstrates the feasibility of consolidating the methodologies introduced in chapters 4 and 6 into a unified system, which exhibits robust performance even under severe conditions.

The limitations of the simulation engine show that there is a large variety of possible improvements for the evaluation setup and the simulation engine itself. In the stationary system simulation, some pedestrians do not move once they have reached a particular position. In the event that a spectator cannot be detected in this particular position, the simulation does not account for any random or natural movements. As a result, in such a case, detection and classification performance will be disproportionately affected. However, the same applies to a spectator remaining in a position where detection and classification work as desired. Consequently, the evaluation of missed detections and correct detections is skewed, and the evaluation is assumed to be balanced.

Furthermore, there are significant improvements that could contribute to the improvement of the camera-based object detection branch of the Bayesian network. As a result of the use of a Faster RCNN approach, anchor design can be refined more efficiently. Since the use case at hand is very specific, certain anchors, either very large or very small, will not be able to contribute to improving the detection quality. Investing more time and effort into the design of anchor bounding boxes is expected to result in a decrease in training effort and an increase in detection accuracy. Additionally, the current anchors are based on the COCO 2017 dataset. They are, therefore, biased toward the objects present in the particular dataset, whereas some of the object classes present in COCO are irrelevant to the present use case. However, improvements in image-based objection detection and classification cannot fully account for impairments of image quality as experienced during extreme weather conditions. The laws of physics apply, and the image quality is degraded in case of rain, nighttime, fog, snow, or the like.

At present, bounding boxes are used after non-maximum suppression of bounding boxes. Considering the anchors before suppression is expected to improve object detection while decreasing the efficiency of the overall sensor detection and fusion due to a higher computational load.

8 Quality Control and Fault Classification in the Manufacturing Industry

In this chapter, the spotlight is on the strategic utilization of specialized domain expertise for precision-oriented data preparation. In many manufacturing scenarios, amassing extensive datasets analogous to a large variety of computer vision applications is an impractical and uneconomical endeavor. Instead, the objective shifts to gathering a high-fidelity, representative dataset that encapsulates the essential classes intrinsic to the specific manufacturing context. Orchestrating such a collection necessitates profound domain knowledge, ensuring the inclusion of sufficient samples from every targeted class. Subsequently, the training procedure for NNs incorporates this expertise, effectively accounting for the implications of misclassification.

8.1 Introduction

The transition to electric mobility has instigated substantial transformations in the automotive sector. This has led to the emergence of new challenges and necessitating modifications to vehicle design and composition. As discussed in [133], these modifications directly influence manufacturing processes, given that electric powertrain components differ from their conventional counterparts. Consequently, the quality of the manufactured parts may be affected due to this lack of experience with novel production techniques.

There has been an increase in the use of innovative technologies in producing electric powertrain elements, such as hairpin welding, intending to increase efficiency and performance. Nevertheless, incorporating these methods can prove challenging and frequently results in the creation of flawed components. There is a significant concern regarding the lack of appropriate supervision and revision techniques for these parts, which warrants attention to ensure the quality and reliability of the final product.

This chapter discusses the development of a technique to detect imperfections in welded hairpins, which are essential components of electric motors, by leveraging expert knowledge during data collection. Multiple network configurations are analyzed and compared using CNNs to categorize quality discrepancies in welding operations. Using both 3D scans and grayscale imagery as input sources, the investigation sought to identify a cost-effective method of identifying defects in the manufacturing process. The approach and the results reported in this section have first been partly published in the Proceedings of the 28th European Signal Processing Conference (EUSIPCO-2020) in 2020 [134].

8.2 State-of-the-Art

Object detection and recognition frameworks have made significant progress in recent years. One-stage and two-stage networks have emerged as the two primary types of architecture. The single-stage networks, such as SSD and YOLO, combine the detection and classification of objects in one step, whereas the two-stage networks divide these tasks into discrete steps. There are several examples of two-stage architectures, including those based on region-based CNNs like R-CNN [47], and its variations, including Fast R-CNN [135], Faster R-CNN [49], and Mask R-CNN [50]. Since hairpin detection is generally regarded as a resolved problem, this investigation focuses primarily on classification.

Identifying production errors as soon as possible reduces the costs associated with their rectification. Quality control is, therefore, of the utmost importance, as is minimizing undetected defects. However, only a few studies in literature implement ML techniques for detecting quality discrepancies in this domain [136, 137]. The use of ML algorithms in the manufacturing of electric motors has also been analyzed in [138]. Only one study has applied ML methods to identify quality deviations in the hairpin welding process [139]. To assess quality deviations, a Charge-Coupled Device (CCD) camera and a CNN were used; however, the accuracy of the network was between 61% and 91%, which is not sufficient for industrial applications.

Three approaches towards classification are discussed in the following sections to increase the accuracy and economics of the quality control process while leveraging domain knowledge for NN architectures and expert-driven data preparation.

1. Custom-developed CNN classifies quality variations in the welding process using grayscale images derived from 3D scans as input.
2. Inception V3 classifies 2D images captured at a 300×300 resolution with a standard industrial grayscale camera.
3. A tailored NN classifies 2D images of a 30×30 resolution with a conventional industrial grayscale camera.

8.3 Methodology

An important step in the production of stators involves the preparation of copper rods into hairpin shapes which are then inserted into a stator lamination stack and connected via laser welding [140]. However, copper's pronounced reflective properties hinder radiation absorption, necessitating increased laser power and leading to characteristic defects during welding [23]. This is why the welding process is thoroughly analyzed to determine common types of errors. Welding classification is segmented into four categories: correct welding (CW), insufficient welding (IW), weld spatter (WS), and weld craters (WC), as depicted in figure 8.1.

During the course of this work, there was no recognized automated system for fault classification. As a consequence, within the existing production workflow, the stator

proceeds through all subsequent processing stages until the final quality control point. Stators that exhibit anomalies in their weld seams are subsequently extracted from the production line, deconstructed, and subjected to manual rewelding. This methodology is both labor-intensive and financially burdensome. To enhance efficiency in the process, early detection of welding flaws is paramount to facilitate targeted rewelding in large-scale manufacturing.

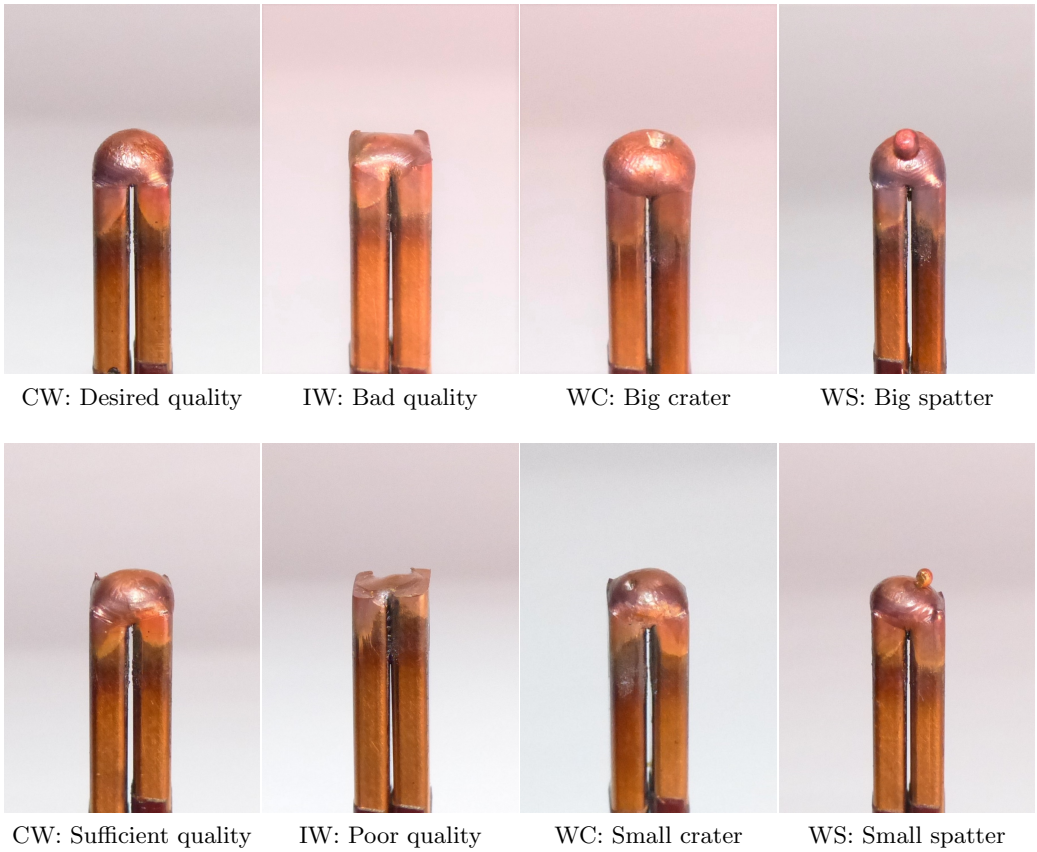


Figure 8.1: Representation of the four quality classes that result from the welding process of hairpins. This figure has been adapted and is based on the original publication [134].

Experimental Setup

The quality of a hairpin is determined by its top part, where a proper connection is essential for electricity transfer. Wire segments, made of bare copper and measuring $100mm$ in length, were placed into a test carrier and welded together at various quality levels.

A Keyence XR-HT40M 3D camera is used to simultaneously capture 3D data and grayscale images of the welding seams. With a 3D camera, inspection is more stable due

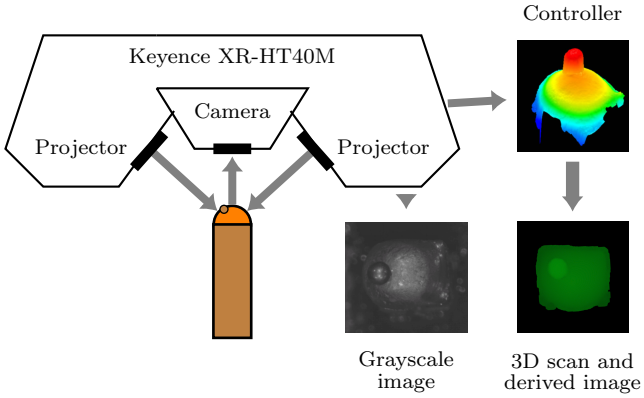


Figure 8.2: The figure shows the experimental setup and is part of the original publication [134].

to the added height information, which provides valuable information for the inspection process. However, the cost of a 3D camera is significantly higher than a conventional 2D camera for industrial applications. Figure 8.2 shows the process of simultaneously capturing 3D scans and 2D grayscale images of a welding spatter (WS), including the resulting 3D scan and the derived image, as well as the grayscale image.

An area containing four hairpins is captured with a grayscale camera and a resolution of $2048x2048$. Hairpins were detected using data from a 3D scanner, resulting in cropped images of hairpins with a resolution of $300x300$. This chapter does not address the topic of object detection in grayscale images. However, various ML-based image processing techniques can be applied to this task [48, 49, 46].

Expert Driven Data Generation

The objective is to create a dataset that encompasses all necessary classes. This task is challenging in an industrial production environment, and generating training samples is costly. A representative dataset is collected and annotated using expert knowledge of defect characteristics. For each class of hairpin welds, 550 to 600 images are manually produced with varying degrees of defect severity. Several data augmentation methods are adopted to further increase the sample count of the training set. These methods include rotations, shifts, and mirroring. As it is not always certain that hairpin images in a production line will be consistently centered and rotated, these techniques are applied to develop a more realistic and diverse training set. To ensure unbiased results, excluding synthetically generated images from the test set is crucial. An overview of the dataset, including data augmentation, is provided in table 8.1.

The dataset is partitioned into training and validation sets, with 80% of the data allocated for training purposes and 20% for validation.

Table 8.1: Division of the dataset for the failure classes

Class	Training set	Test set	Sum
IW	456	104	560
WS	455	102	557
WC	438	125	563
CW	478	126	604
Sum	1,827	457	2,284
Augmented	91,350	-	-

3D Data Analysis

3D scans undergo preprocessing to improve the accuracy and efficiency of the classifier. The preprocessing workflow consists of the following steps:

- The 3D scan is first centered around the hairpin and cropped to dimensions of $450x450$ pixels. This step is illustrated in figure 8.2 using a raw 3D data sample. The height data spans from 0 to 16 millimeters, as depicted in figure 8.3.

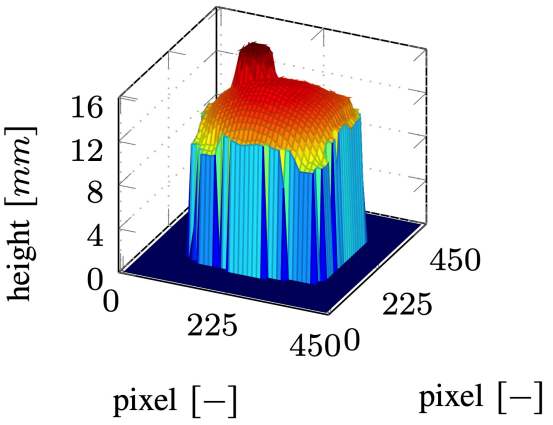


Figure 8.3: This figure displays the height transformation of 3D data, as featured in the original publication [134].

- Next, the 3D scan undergoes compression in both the x and y directions, resulting in a resolution of $30x30$ pixels. Simultaneously, the height range is adjusted to fit values between 0 and 255.
- From the refined 3D data, a grayscale image is generated.
- Finally, the image undergoes normalization. This is achieved by deducting the mean pixel value from the cumulative pixel value of the entire dataset and subsequently dividing the outcome by the standard deviation, as shown in figure 8.4.

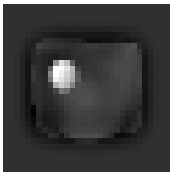


Figure 8.4: Normalized grayscale image derived from 3D data.

30x30 Grayscale Images Derived From 3D Data Based on Tailored Network Architecture

A specialized model designed for images derived from 3D data is introduced, which incorporates convolutional blocks (Conv-Blocks). Each Conv-Block is composed of a series of convolutional layers, followed by batch normalization, a ReLU activation function, and a pooling layer. In the suggested architecture, the NN employs four such Conv-Blocks, with the filter count doubling for each subsequent block. The initial Conv-Block features two convolutional layers, each equipped with eight $3x3$ pixel kernels. The subsequent second, third, and fourth blocks maintain this configuration but with a kernel count that is double that of their predecessor. Notably, the fourth block is configured for global average pooling instead of the typical maximum pooling.

Additionally, the model includes a fully connected layer comprising 32 neurons. This is followed by a batch normalization layer, a ReLU activation function, and a dropout layer set at a rate of 0.5. The concluding layer is designed with a neuron for each class, utilizing the softmax activation function to yield the final output likelihood.

The optimal training parameters are determined using a stochastic gradient descent algorithm with a batch size of 150. The step size is determined to be $1e-3$, which reduces by 0.9 every five training epochs. In addition, this optimization method incorporates a Nesterov momentum of 0.9. Categorical cross-entropy is used as a cost function.

A class weighting function is implemented as part of the cost function to increase or decrease the weighting of specific classes during training. This assigns greater weight to the cost of misclassifying faulty welds, resulting in larger gradients and a greater impact on the model. The distinct classes are weighted as follows:

$$\left[w_{CW},\, w_{IW},\, w_{WS},\, w_{WC}\right] = \left[1,\, 10,\, 10,\, 10\right]$$

The weights are derived based on process knowledge and imply that misclassifying the faulty classes is ten times the cost of misclassifying the classes of sufficient quality.

Grayscale Data Analysis

The acquisition of monochromatic image data is concurrent with the acquisition of 3D data, allowing a fair comparison of the methodologies. Hairpin crops are normalized and resized to meet the input resolution requirements for the NNs. Two networks based on monochromatic data are evaluated to determine a well-fit network architecture: Inception V3 is selected as a baseline classifier due to its proven success in various transfer

learning applications, as demonstrated in [141]. In this case, feature extraction is customized to accommodate the specific needs of the use case, using an input size of 300×300 pixels.

In addition, a proposed custom network architecture based on the same images but with a reduced resolution of $30x30$ pixels is developed and trained from scratch, addressing the particular requirements of the use case.

300 x 300 Grayscale Image Analysis Based on Inception V3

Inception V3 is selected as the baseline network architecture, owing to its enhanced efficiency, achieved by decomposing sizable convolution kernels into smaller counterparts and minimizing the parameter count [142]. Weights from Inception V3, pre-trained on ImageNet data, are utilized for the current use case. Even though the images in this case differ from those in ImageNet, these weights provide a foundational starting point. The process of extracting features from images has inherent similarities, which means the weights and feature extraction can be fine-tuned according to the unique challenge during the training phase. The network incorporates all layers of Inception V3 up to the mixed 10th layer. This is succeeded by three fully connected layers and a concluding softmax layer. The initial two fully connected layers have $1,024$ neurons each. They are followed by another layer containing 512 neurons, and ultimately, the softmax layer offers four output classes. After the fully connected layer, there is a ReLU activation and a dropout layer with a 0.1 ratio. Given that the default ImageNet weights are not suitable for extracting the necessary features for the specific scenario, all layers undergo retraining. However, starting with the pre-trained weights lessens the training effort compared to using randomly initialized weights.

The utilized loss function is the categorical focal loss with the parameters $\alpha = 0.25$ and $\gamma = 2$ [111]. The training is orchestrated by the Adam optimizer, with a learning rate of $lr = 5 \cdot 10^{-6}$ during the initial training phase and $lr = 1 \cdot 10^{-6}$ during the subsequent phase, using accuracy as the evaluation metric [110]. The hyperparameters $\beta 1$ and $\beta 2$ are set to values $\beta 1 = 0.9$ and $\beta 2 = 0.999$, respectively. Results are derived from a five-fold cross-validation approach.

The training process is divided into two phases: the primary phase leverages pre-existing weights, and the latter focuses on the specific nuances of the application. The initial learning phase spans 100 epochs with a comparatively higher learning rate, while the second phase consists of 50 epochs. For both phases, the batch size is set at 32 and the steps per epoch at 100.

Class weights are selected based on domain expertise to mitigate class imbalances and associated costs of errors.

$$\left[w_{CW},\, w_{IW},\, w_{WS},\, w_{WC}\right] = \left[1,\, 0.4,\, 0.75,\, 0.35\right]$$

30 x 30 Grayscale Image Analysis Based on Tailored Network Architecture

The proposed network architecture for monochromatic images has an input size of 30×30 pixels and is partitioned into three primary components. The initial component comprises five convolutional layers, followed by batch normalization, ReLU activation, and a max-pooling layer. The kernel sizes used in this component are 1×1, 3×1, 1×3, 1×1, and 3×3, with a stride of 1. This component functions as a feature extractor, comparable to a dense block as utilized in [89]. A stride of 2×2 is included in the maximum pooling.

The second component is a convolutional layer with a 3×3 kernel, succeeded by batch normalization, activation, and max pooling layers. This component aims to further refine and reduce the feature space. The third component of the network includes two fully connected layers, one with 1024 neurons and the other with 512 neurons, as well as a softmax activation layer.

In addition to batch normalization and ReLU activation, a dropout of 0.1 is applied after the first two layers. Weights are initialized with the Glorot initializer [143], and all convolutional layers employ 32 filters. The training process utilizes a focal loss and an Adam optimizer with a learning rate of $lr = 10^{-4}$ and a similar parameterization to that used for re-training the Inception V3-based architecture.

The training is executed in two phases, each comprising 100 epochs, with batch sizes of 32 and 100 steps per epoch. The initial phase treated all classes uniformly, while the second phase incorporated class weights equal to the ones used for the 300×300 grayscale images.

8.4 Evaluation of Experimental Results

All network architecture results are compared based on identical metrics in this section. The proposed models are implemented in Keras version 2.2.4 using the TensorFlow backend version 1.14.0.

Training Process

The training process adopts the five-fold cross-validation technique. When utilizing grayscale images derived from 3D data as input, the training achieves high accuracy and stability, displaying minor variations and no discernible overfitting or underfitting.

As highlighted in section 8.3, the training for the conventional grayscale images follows a dual-stage strategy while also applying a five-fold cross-validation. No signs of overfitting or underfitting are observed in either the inception-based network or the bespoke network architecture.

Confusion Matrix

The confusion matrix (CM) resulting from the five-fold cross-validation is shown in figure 8.5.

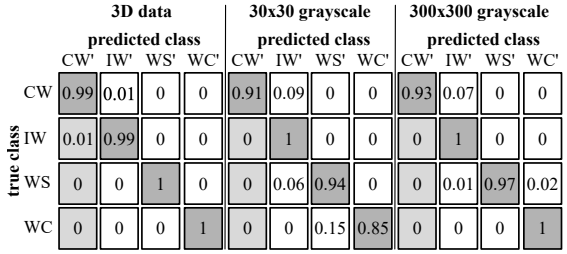


Figure 8.5: Visualization of the classification results based on relative values. Left: Confusion Matrix (CM) with 3D data. Middle: CM with $30x30$ grayscale data using the proposed network architecture. Right: CM with $300x300$ data using the Inception V3-based architecture. This figure is part of the original publication [134].

The CMs reveal that the trained models utilizing 3D data and monochromatic images effectively differentiate the classes, as evidenced by the high values on the CM diagonals (highlighted with a darker gray color). Entries not on this diagonal signify incorrect classification by the model.

Evaluation Metrics

As a result of the confusion matrix, several evaluation metrics are calculated that quantify the ability of the models to classify data. This includes classification accuracy, precision, recall, and F_β-score. Top-1 and Top-5 accuracy is utilized for comparison in most image classification tasks. The use of the Top-5 metric is not suitable for critical image classification applications. Precision and recall are equally balanced by the F1-score, indicating that all error types have equal costs. Nevertheless, in this particular case, the cost of misclassifying a fault-free sample as defective is significantly lower than the cost of failing to detect a fault. Consequently, a F_β with a $\beta = 3$ is selected for the comparison to emphasize the importance of the cost differences:

$$F_\beta = (1+\beta)^2 \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{8.1}$$

Table 8.2 lists the classification metrics of the three implemented networks.

As demonstrated in table 8.2, the accuracy for the approaches relying on 3D data and monochromatic images of 300×300 pixels both exceed 99%. The accuracy of the

Table 8.2: Accuracy, Precision, Recall and F_β-score

Type	Overall Accuracy (%)	Average Precision (%)	Average Recall (%)	Average $F_{\beta=3}$-score (-)
3D Scan	99	99	99	1.59
Image (300x300)	99	99	99	1.58
Image (30x30)	94	96	92	1.48

algorithm decreases by approximately 5% when using monochromatic images of dimensions 30×30 pixels in conjunction with the tailored NN. According to these findings, the proposed CNN architectures are capable of detecting welding defects accurately. Reliable production systems need to have a high recall rate. The proposed model utilizing 3D data and 300×300 pixel monochromatic images left 1% of failures undetected or categorized those samples as other failures. The recall is significantly reduced when monochromatic images of 30×30 pixels are utilized.

8.5 Conclusion and Outlook

The results provide evidence that it is possible to address hairpin welding quality control and that satisfactory results can be achieved. To train NNs, input data from a 3D scanner and monochromatic images are utilized. Classifications based on 3D scanner data or large monochromatic images attained comparable accuracy and F_β-scores, with $\beta = 3$. Despite marginally worse performance than the other two networks, the tailored NN architecture based on 30×30 monochromatic images still demonstrates promising results given the considerably reduced network and image sizes compared to the Inception V3-based approach. Additionally, the network is trained from scratch and only trained on the collected data samples without leveraging any transfer learning. This underscores the importance of leveraging domain expertise in both data collection and network architecture design to ensure a more efficient use of resources during data collection, training, and inference.

Effective class separation is achieved by both feature spaces: the one derived from 3D information and the one constructed from monochromatic images. A significant advantage of camera-based systems over 3D scanners is their reduced cost, particularly in an industrial setting where lower quantities or more frequent modifications to the manufacturing process might be necessary due to new production technologies and processes. It is sufficient to make only software modifications in such situations, assuming a 2D camera is used to resolve a new problem.

Future work will focus on integrating the developed solutions into manufacturing processes and assessing their performance using state-of-the-art 3D scanners. Based on the outcome, the developed networks may need to be enhanced. The acquisition of additional training data may also provide an opportunity for better network generalization prior to its integration into production. The approach of harnessing domain-specific knowledge for data preparation and selection not only validates the findings from previous chapters but also offers potential for adaptation in other fields across industries.

The significance and potential of targeted data collection are vital for scenarios where data gathering is costly. The results obtained through transfer learning and even constructing a NN from the ground up highlight the promise of this method in minimizing data requirements and, consequently, expenses. Furthermore, the implications of these findings on effective data usage will play an important role in shaping future research and advancements across different industries.

9 Boundary Enhanced Semantic Segmentation

A semantic segmentation technique is presented throughout this chapter that is tailored for high-resolution Scanning Electron Microscope (SEM) images, intended for detecting hardware Trojans and fake ICs. A NN architecture is developed using domain-specific expertise, and pre-trained encoders are implemented to address the limited availability of training data. An expert domain knowledge-based segmentation network that integrates a boundary stream to prioritize separating technological features is proposed. The effectiveness of the proposed methodology is assessed through a comparative analysis involving the proposed approach, a baseline method, and two advanced, high-performing segmentation networks. The general approach and the results reported in this chapter have first been partly published in the Proceedings of the 30th European Signal Processing Conference (EUSIPCO-2022) in 2022 [144].

9.1 Introduction

Semiconductor devices must perform securely in critical applications, as found in the medical or automotive industry. Electrical testing and process control are important parts of the manufacturing process for integrated devices. A multi-billion dollar industry has resulted from counterfeit electronics infiltrating the market through horizontally distributed supply chains [145]. Using SEM images of semiconductor device cross-sections, a novel approach is proposed, building on well-established analytical procedures. A method such as this may be used to characterize internal processes, defect analysis, root-cause analysis, or as a tool to facilitate future techniques for counterfeit detection.

A critical aspect of understanding the current technology and production process is assessing the distance between the technological features of microchips. Whether these features are accurately categorized directly impacts the quality of applications that use the segmentation for additional processing. This chapter proposes a tailored modeling architecture incorporating properties of the captured images and expert domain knowledge to address the unique challenges SEM images present.

The examined SEM images encompass metal layers and VIAs, with fields of view ranging from 4 µm to 70 µm. For the purpose of minimizing variations within the dataset, only images with metal layer to VIA ratios of the same magnitude were selected. With two microscopes, all images were captured at a resolution of 1024×768 pixels or 1280×960 pixels. As supervised semantic image segmentation requires annotated SEM images, there is a substantial limitation on the number of images available, resulting in a limited

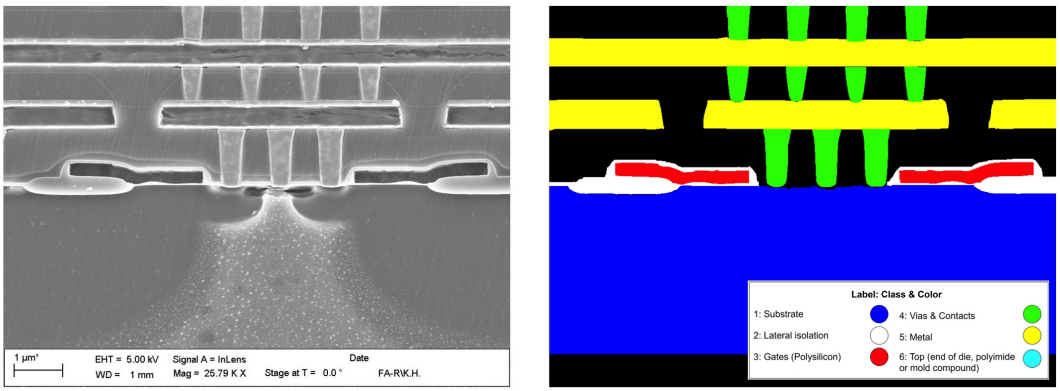


Figure 9.1: The figure shows a SEM image on the left and its pixel-wise labeling on the right. The figure is based on the corresponding original publication [144].

dataset. It is, therefore, necessary to address overfitting, class imbalances, and optimization challenges. As illustrated in figure 9.1, seven classes were identified as most relevant for semantic segmentation.

Automating reverse engineering tasks based on IC structures captured as an image is difficult. In the preparation of the sample, the visual appearance of the cross-section can vary due to manual cutting and the use of chemicals. Damage occurs to the ICs when cut, and there is a substantial variation among the probes. Electron microscopes are used to acquire images iteratively at increasing magnifications. A SEM image shows bright boundaries as a result of electron reflection at edges, while surfaces of a component appear homogeneous, as shown in figure 9.1. As a result, the appearance of objects in this method is substantially different from that in most semantic segmentation methods.

9.2 Related Work

During the last few years, ML techniques have gained increasing prominence in hardware security, primarily as a means of protecting against hardware Trojans and counterfeit ICs. Another use case for ML is Physical Unclonable Functions (PUFs) models, which help identify and avoid IC overbuilding or tampering [146]. Reverse engineering of hardware utilizes ML models primarily for analyzing IC layout images. The use of both supervised and unsupervised learning approaches has been reported in numerous papers for identifying material layers and standard cells and detecting malicious modifications to layouts [147]. One example is applying a fully convolutional network with a VGG-16 encoder to segment metal tracks and VIAs, as described in [148]. Another approach is using an unsupervised K-means method for the same task; however, limitations arise from image preparation and variations, as reported in [149].

Traditional methods for identifying counterfeit products have mainly focused on analyzing packaging, as demonstrated in studies such as [150], [151], and [152]. These papers present various computer vision-based techniques to distinguish between genuine and counterfeit ICs. This work introduces a novel approach enabling reverse engineering automation at the technology level.

This chapter utilizes the U-net architecture as a reference architecture because it has proven effective for small datasets across multiple domains. One example from the medical field where U-Net worked well is the ISBI cell tracking challenge 2015 [60]. Other segmentation architectures have been developed in recent years, such as the FPN and the PSPNet. A more detailed overview is given in chapter 2. FPN, employing a pyramidal hierarchy of multiple scales, is a region proposal and classification network that generates a feature pyramid and has demonstrated superiority over other region proposal networks like DeepMask, SharpMask, and InstanceFCN [63]. PSPNet enhances feature representation by extracting features at multiple scales and has exhibited exceptional performance in the ImageNet scene parsing challenge 2016 [61]. The GSCNN is a network employing a two-stream architecture to incorporate shape information into an additional stream alongside the traditional CNN feature stream [65]. This architecture has shown superior performance in automotive segmentation use cases.

9.3 Proposed Methodology

Inspection Framework

An analysis of the internal processes of IC manufacturers is necessary, including validating process stability, detecting defects, root-cause analysis, and identifying counterfeit ICs. The automated reverse engineering process is shown in figure 9.2, starting with IC preparation, resulting in advanced analysis and insights. The chip is cut vertically, polished, and prepared as part of the processing steps. SEM images are captured at various magnification levels to inspect the relevant areas of the chip required for technology determination.

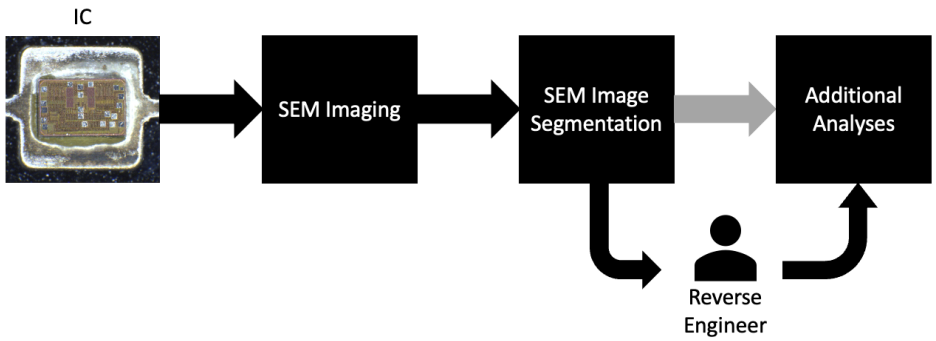


Figure 9.2: Automated IC framework for SEM image segmentation enabling advanced analyses with humans in the loop. The figure is is based on the corresponding original publication [144].

This work presents an approach for automating the reverse engineering process by segmenting SEM images, forming the basis for successful process execution. However, it is crucial to note that a reverse engineer must assess the image segmentation quality to determine if further analysis can be based on the results.

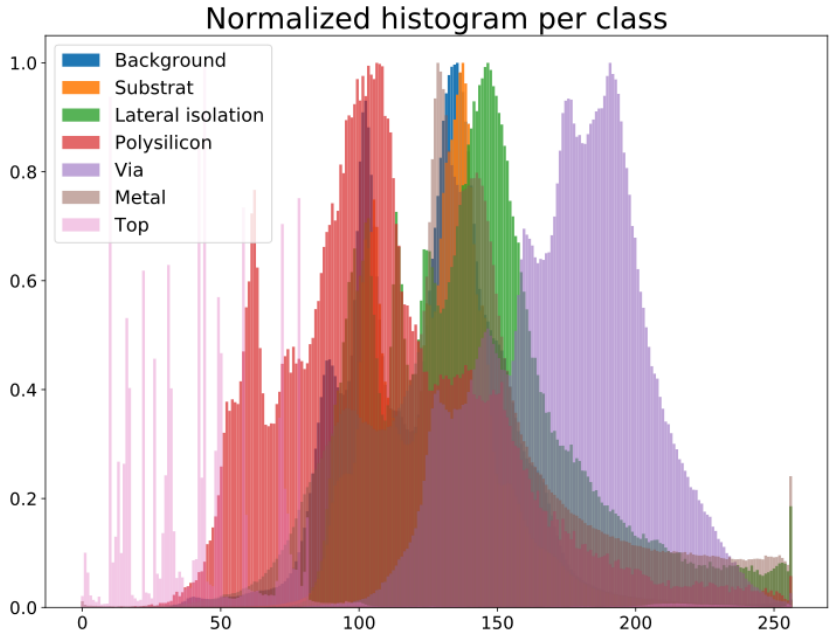


Figure 9.3: Normalized histogram of gray values per class

Dataset

SEM images are saved as grayscale 8-bit images with a resolution of either 1024×768 or 1280×960 pixels. In figure 9.1, the seven classes relevant to the intended semantic segmentation use case are shown. The classes are substrate, lateral isolation, polysilicon, VIA, metal, top of the die, and background. A significant imbalance exists among the classes due to the limited dataset and the structure of the ICs. The majority of all pixels are background (38.1%) and substrate pixels (29.6%), while lateral isolation (1.3%), polysilicon (0.5%), and VIA (2.5%) are less frequent. It depends on the investigated technology, how these classes are arranged, and how frequently they occur. Different material densities can be attributed to the various shades of gray in the images. Investigating the normalized histogram of grayscale values, as shown in figure 9.3, emphasizes that segment separation cannot rely on grayscale values alone, necessitating more advanced techniques. Data specifically labeled for the current use case is required for the highly specialized task of segmenting SEM images. It is, however, extremely resource-intensive to generate ground truth for this type of image due to the requirement of expert domain knowledge for annotation. For the present work, a pixel-level accurate approach was chosen. It is more time-consuming and expensive than polygon-based annotation but avoids incorrectly labeled pixels. This way, 40 images were manually labeled to ensure high-quality training data, enabling a more robust training process.

Pre-Processing

In the application of ML, preprocessing data is an essential and fundamental step. In the case of NNs, a high-dimensional feature space is employed to distinguish between classes. It has been theoretically demonstrated that discrimination between classes becomes easier as the dimension of the feature space increases. There is a direct relationship between the number of layers and nodes in the network and its learning ability. Increasing the number of layers and nodes increases the learning capacity of the network, allowing it to learn more complex transfer functions. When assessing the learning capacity of a model, it is important to consider the concepts of overfitting and underfitting in the context of the use case. To simplify the transfer function that the NN must approximate, domain knowledge from experts is leveraged. To this end, the data is pre-processed before it is fed into the network. Throughout this chapter, images are resized to 224×224 pixels to enhance feature transferability and reduce memory requirements. This step aims to minimize training time and manage computational resources as effectively as possible. Transfer learning is leveraged for the present approach by using pre-trained encoders. However, three stacked grayscale images are used instead of RGB data as input.

Baseline Classifier

Data Augmentation

An important aspect of enhancing the generalizability of a model is the use of data augmentation instead of increasing the amount of data to achieve this goal [110]. Augmentation generates synthetic training data by making minor modifications to the original images. Numerous data augmentation techniques exist, each offering its own set of strengths and weaknesses. For this specific application, the following methods were assessed: optical distortion, grid distortion, elastic transformation, median blurring, Gaussian noise injection, adaptive histogram equalization (CLAHE), random cropping, and horizontal flips. Vertical flips are excluded due to the unique properties of SEM images and assumptions concerning their structure and intersegment relationships. The knowledge graph capturing the relationships among classes is shown in figure 2.21. With a probability of 0.5, these techniques were applied to the training data. Crops and distortions were randomized, whereas all other techniques used the default hyperparameters. The dice score, which measures the similarity between target A and output B, serves as the performance metric used to evaluate the techniques individually leveraging the baseline classifier:

$$Dice \quad score = \frac{2\,|A \cap B|}{|A| + |B|} \tag{9.1}$$

This work utilizes the dice coefficient as a performance metric, accounting for false positive class detection and proving especially useful in situations involving imbalanced datasets. Data augmentation technique results are depicted in figure 9.4. Notably, neither Gaussian noise injection nor CLAHE preprocessing led to performance improve-

ments. As a result, all the techniques mentioned above, except those two, were implemented.

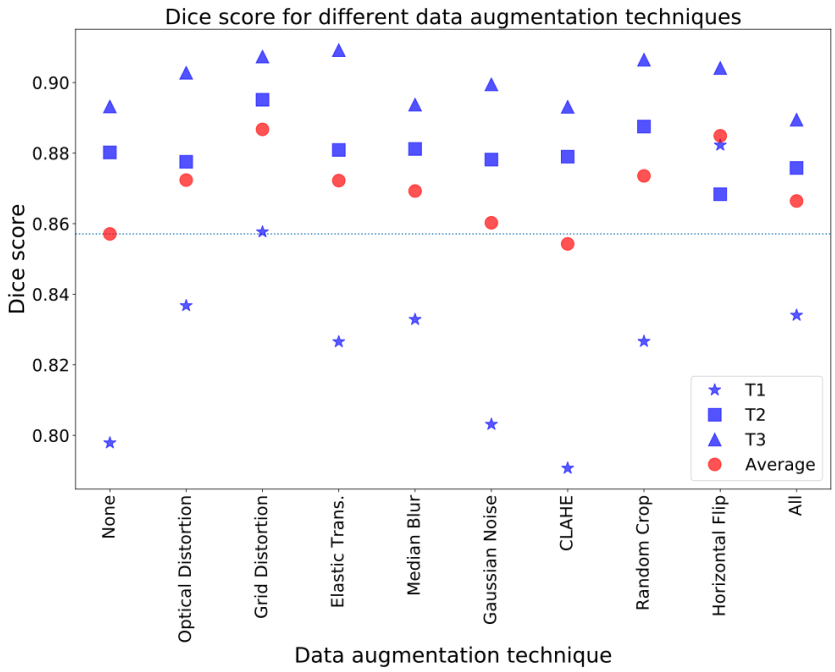


Figure 9.4: The influence of distinct data augmentation techniques on network performance is examined, with three-fold cross-validation applied to evaluate each technique, producing three splits: T1, T2, and T3. Additionally, the average performance is depicted in red, and a dotted line indicates the average dice score of the baseline without data augmentation.

Encoder Selection

Given the limited data available for this specific application, the dataset is considerably smaller than those typically encountered in supervised learning tasks. Training a classifier from scratch would result in overfitting to the training data and poor performance. This issue is addressed using transfer learning, which enables the use of pre-trained feature extractors and their adaptation to the task at hand. A pre-trained feature extractor, the encoder, is enhanced by training on SEM images while preserving feature patterns as found in the dataset used for pre-training. This approach has been documented in literature [153].

The proposed method employs encoders, which are initially trained on the ImageNet dataset [154], and subsequently fine-tuned for the application of SEM image segmentation. This method is based on the hypothesis that adapting a pre-trained encoder to a specific problem through transfer learning and data augmentation is advantageous. The unavailability of pre-trained encoders for this application further motivated the selection of the ImageNet dataset, which excels at capturing semantic image information.

Previous research has demonstrated that NNs employing transfer learning exhibit superior performance and faster convergence than those trained from scratch. This is particularly relevant for this application, as it involves a limited dataset, as noted in [153]. A high-performing encoder is required for a high-quality segmentation, as reported in [155, 156]. In this chapter, various encoders are examined and trained using the 2012 ILSVRC ImageNet dataset [154], including ResNet18, ResNet50, ResNet101, ResNet152, SE-ResNeXt50, SE-ResNeXt101, VGG11, VGG19, Densenet161, Densenet201, and DPN131.

A constant learning rate of 1e-4 was employed during the evaluation, and identical training and validation splits were used for all encoders due to the three-fold cross-validation. Performance was assessed using the dice score, and the number of parameters for each encoder was considered an approximation for the network size. SE-ResNeXt50 encoder exhibited the best performance and is subsequently used throughout this chapter. The evaluation outcomes are displayed in figure 9.5 while referring to the cross-validation dataset splits as T1, T2, and T3.

A noteworthy observation concerns the relationship between the number of parameters and the measured peak performance. Encoders with fewer parameters demonstrated improved performance when employed with a PSPNet network, suggesting that overfitting remains a concern for larger networks. The SE-ResNeXt50 achieved the highest average performance, while DPN131 had the lowest average performance and the greatest number of parameters. This indicates that shallower and narrower networks are more suitable for smaller datasets, even though deeper and wider networks can tackle more complex tasks. This finding aligns with observations reported in [157] and stresses the importance of leveraging domain knowledge to reduce network complexity.

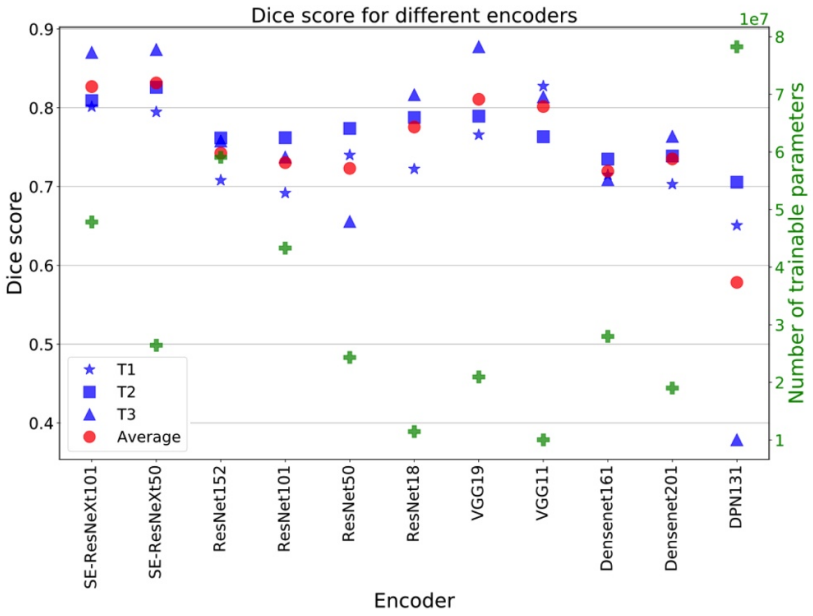


Figure 9.5: Influence of the used encoder architecture on the network performance

Network Architecture Evaluation

A crucial factor in selecting the appropriate architecture and encoder is the ability to compare results against a benchmark. Initially, U-net, FPN, and PSPNet are selected for comparison, and for assessment, the same encoder is utilized for each network architecture. The outcomes are depicted in figure 9.6. Pyramid and multi-scale networks perform better, with FPN and PSPNet demonstrating similar results. PSPNet, however, has a lower standard deviation and, as a result, is chosen for the following steps due to its higher average performance.

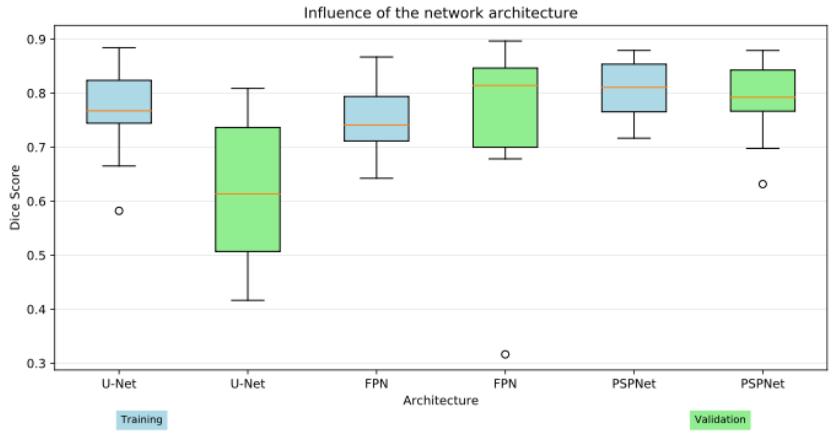


Figure 9.6: Influence of the used upsampling architecture on the network performance

Batch Size

Batch size is a critical determinant of computational resource requirements, generalization capabilities, and training dynamics. Larger batch sizes facilitate parallelization during training and offer a more accurate gradient estimate [158]. However, memory requirements also increase linearly with batch size, which constrains numerous applications. As per [159], employing a smaller batch size can enhance generalization capabilities while reducing memory requirements. It is evident from the evaluation of batch sizes between 1 and 32 that a batch size of four results in satisfactory performance, whereas smaller and larger batch sizes negatively affect performance. With a restricted dataset, a small batch size can impede effective network learning, and a larger batch size can contribute to overfitting.

Optimizer

Several optimization algorithms are available, most based on stochastic gradient descent. However, no universally optimal algorithm exists. As a result of its relative robustness regarding hyperparameter selection, the Adam optimizer was chosen for this application [110]. A key feature of the Adam optimizer is that it incorporates adaptive estimations

for the first and second moments of the gradient. In addition, it employs bias corrections for the first- and second-order moments to enhance stability. The default parameterization, as presented in the original paper [160], is adopted for the present use case. Given the challenges introduced by the small dataset, cross-entropy is used as a loss function because of its ability to address the class imbalance by weighting the channels. Additionally, cross-entropy results in smoother gradients compared to the dice loss.

Knowledge Exploitation for Boundary Enhancement

The process of IC manufacturing and reverse engineering, from IC preparation to analysis, follows a defined set of steps. Expert knowledge of these processes is utilized to enhance automated segmentation. This is done in two ways: Firstly, capturing SEM images of ICs has the drawback of creating blurred edges due to electron reflection; this is overcome by improving boundary detection. Secondly, the manufacturing process of microchips is controlled and follows specific patterns; these patterns provide boundary conditions that are used to regulate the learning process of the segmentation. These conditions are captured in the knowledge graph as depicted in figure 2.21.

The use of CNNs poses a significant challenge due to the loss of spatial resolution, which results in the suppression of high-frequency components and, as a result, blurred edges in segmentation. Using techniques such as skip connections and feature map concatenations, it is possible to preserve more high-frequency components while only adding a small amount of overhead by passing additional components through the entire network. A downside of this is the increased inefficiency of the CNN caused by the growing complexity of the segmentation problem, as described in [65]. For this application, robust boundary detection is critical. However, this task presents significant challenges due to the presence of blurring and noise in SEM images, particularly at IC edges.

GSCNN

A two-stream architecture is adopted for semantic segmentation, wherein shape information is processed through a separate branch in the network. By focusing on shape information in this branch, a more accurate boundary detection can be achieved, for which high-frequency components are vital. The utilization of a distinct shape stream is elaborated upon in [65]. Additionally, the authors show that cross-entropy alone is insufficient for training such a NN architecture. Segmentation and boundary predictions must be jointly supervised, necessitating two separate predictions. Consequently, cross-entropy is used for the segmentation map, while binary cross-entropy is applied to the boundaries, resulting in a combined loss, $\mathcal{L}_{combined}$:

$$\mathcal{L}_{combined} = \lambda_1 \mathcal{L}_{\mathcal{BCE}}\left(s, \widehat{s}\right) + \lambda_2 \mathcal{L}_{\mathcal{CE}}\ \left(\widehat{y}, f\right) \tag{9.2}$$

The Sobel filter in the x and y directions are utilized as the ground truth boundary maps, $\widehat{s} \in \mathbb{R}^{HxW}$, and $\widehat{y} \in \mathbb{R}^{HxW}$ is the ground truth for the semantic map, with a width of W and a height of H. The authors of [65] suggest setting the hyperparameters λ_1 and λ_2 to λ_1=20 and λ_2=1, which is adhered to in this work. A significant constraint

of the GSCNN network is the extensive number of parameters that need training, a result of utilizing the ResNet encoder. An alternative, streamlined version of GSCNN, named L-GSCNN, is assessed. This variant has fewer parameters because it employs SE-ResNeXt50 as its encoder. Evaluations were conducted on both GSCNN and L-GSCNN.

Topological Regularization

During the iterative IC production process, class relationships are represented using a knowledge graph that captures the interconnection of classes. A topological regularization component $\mathcal{L}_{TR}$ is added to the loss function to improve the network. This component penalizes class transitions that are not present in the knowledge graph, as shown in figure 2.21. The transition of two neighboring pixels of the same class is not penalized. The regularizer from equation 9.3 only penalizes invalid vertical class transitions, as the boundary conditions introduced by the manufacturing process mainly occur in this direction. Pixel $p_{i,j}$ is compared to the pixel $p_{i,j+1}$ above it. When an invalid transition occurs, the function $f_{boundary_{violation}}$ returns one, increasing the regularization weight. The number of pixels normalizes the regularization loss by multiplying width W and height H.

$$\mathcal{L}_{TR} = \frac{\sum_{i=0}^{W} \sum_{j=0}^{H-1} f_{boundary\_violation}(p_{i,j}, p_{i,j+1})}{W \times H} \tag{9.3}$$

The combined and regularized loss $\mathcal{L}_{combined\_reg}$ is determined as follows:

$$\mathcal{L}_{combined\_reg} = \mathcal{L}_{combined} + \lambda_3 \mathcal{L}_{\mathcal{TR}} \tag{9.4}$$

Applying the combined and regularized loss is intended to simplify the model, which addresses the limited data availability while reducing the complexity. However, the regularization captures an essential part of the boundary conditions, but not all of them. Additionally, the available data could introduce undesired biases due to the occurrence frequency of certain class transitions. Applying the loss throughout the entire training process could result in overfitting. Consequently, regularization is only applied during the first 15 epochs of the training to exploit the domain knowledge on transitions while not risking getting stuck in local minima.

PUBNet

A specialized network architecture referred to as PSPNet with U-Net-like upsampling and boundary enhancement (PUBNet), as depicted in figure 9.7, is proposed to address the limitations of prior networks and enhance the current use case. Upsampling the low-dimensional feature map outputs of PSPNet in a single step would result in information loss. Such an approach would negate the advantages of employing a separate shape stream. To circumvent this issue, an upsampling technique akin to U-Net is implemented. Consequently, the expansive path of the network encompasses 3x3 convolutional, batch normalization, and ReLU layers. 2x2 transposed convolutions with a stride of two are used for upsampling. Through skip connections, the outputs of each upsampling stage

are concatenated with the corresponding feature maps from the encoder. This network is based on a PSPNet architecture that employs an upsampling method similar to U-Net, while also integrating a distinct shape stream to preserve boundary information. To ensure comparability, SE-ResNeXt50 is utilized as the encoder.

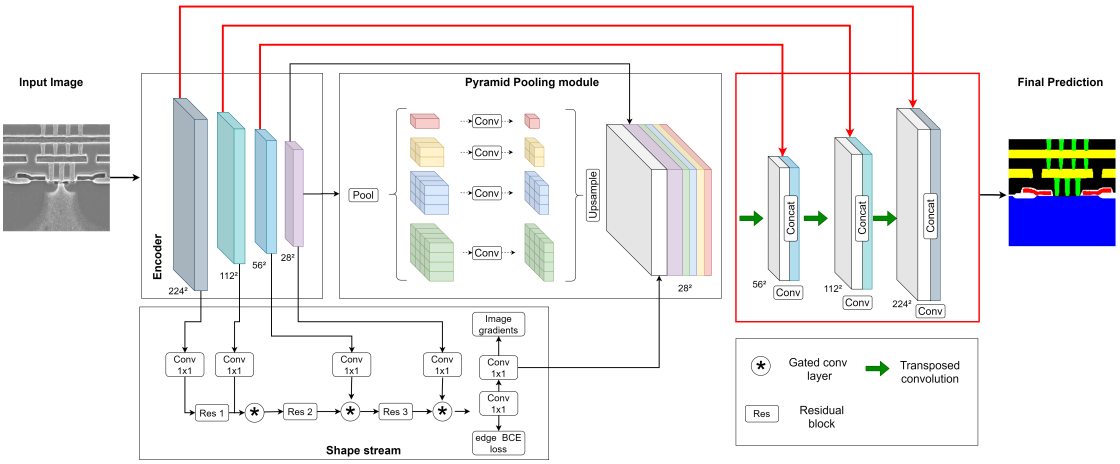


Figure 9.7: PUBNet: The outputs of each stage of the encoder are passed forward to the shape stream and to the corresponding level in the upsampling path (highlighted in red). The size of each feature map is indicated below each feature map. The figure is based on the corresponding original publication [144].

9.4 Results

A comparative analysis of various architectures specifically targeting boundary detection was conducted, utilizing a PSPNet with a SE-ResNeXt50 encoder as the baseline. A summary of the results is shown in table 9.1. Compared to the baseline, all approaches enhanced the performance. Compared with GSCNN, L-GSCNN exhibits a smaller outcome variance, suggesting that fewer network parameters are more suitable for this application. As an additional metric to the dice score, the assessment includes mean intersection over union (mIoU), yielding similar findings and affirming the trends observed in the dice score. A qualitative segmentation evaluation was also performed by examining SEM images compared to the corresponding ground truth. The results revealed that both the baseline network and L-GSCNN were incapable of distinctly separating fine-grained segments, whereas PUBNet successfully achieved clear separation.

9.5 Conclusion and Outlook

This chapter presents a method enabling automated analysis and detection of counterfeit semiconductor devices, surpassing conventional superficial packaging examination. The results indicate that the proposed network architecture excels compared to existing methods for this specific application. This is achieved by integrating techniques found

Architecture	Parameters	Dice Score				mIoU			
		T1	T2	T3	Avg	T1	T2	T3	Avg
Baseline	26 331 511	0.852	0.855	0.840	0.849	0.372	0.361	0.355	0.363
GSCNN	137 275 118	0.889	0.850	0.927	0.889	0.38	0.366	0.439	0.395
L-GSCNN	42 525 534	0.910	0.876	0.929	0.905	0.415	0.373	0.448	0.412
PUBNet	31 073 009	0.894	0.917	0.91	**0.907**	0.429	0.416	0.42	**0.422**

Table 9.1: Different shape enhancing architectures evaluated based on their number of parameters, dice score, and mIoU shown per training run (T1, T2, and T3) and average (Avg) performance.

in medical imaging and AD while exploiting domain knowledge on the manufacturing process to match the requirements of the present use case. The network effectively separates individual components by prioritizing image boundaries through the use of a distinct shape stream, hence, leveraging expert domain knowledge. The results of these segmentations can be employed for further comprehensive analysis, including detecting hardware Trojans and counterfeit devices. This approach, leveraging domain-specific knowledge, holds the potential for expansion to other fields and confirms findings from previous chapters.

The current work is constrained by the classes represented in the dataset and the available training data. The data for some classes is severely limited, resulting in poor performance for underrepresented classes. Classes absent from the dataset while relevant for defining semiconductor device technology include deep trench geometries, package characteristics, and gate oxide geometries. Moreover, variability between different microscope imaging settings and preparation techniques has not been taken into consideration. As a result of the limited availability of annotated images, a more thorough evaluation is not possible.

Future research should focus on designing experiments that facilitate counterfeit detection at the technological level. Various imaging technologies, such as optical or transmission electron microscopes, could be incorporated into the proposed method. To improve the results, generating an increased number of labeled images is necessary, ensuring that all relevant classes are adequately represented.

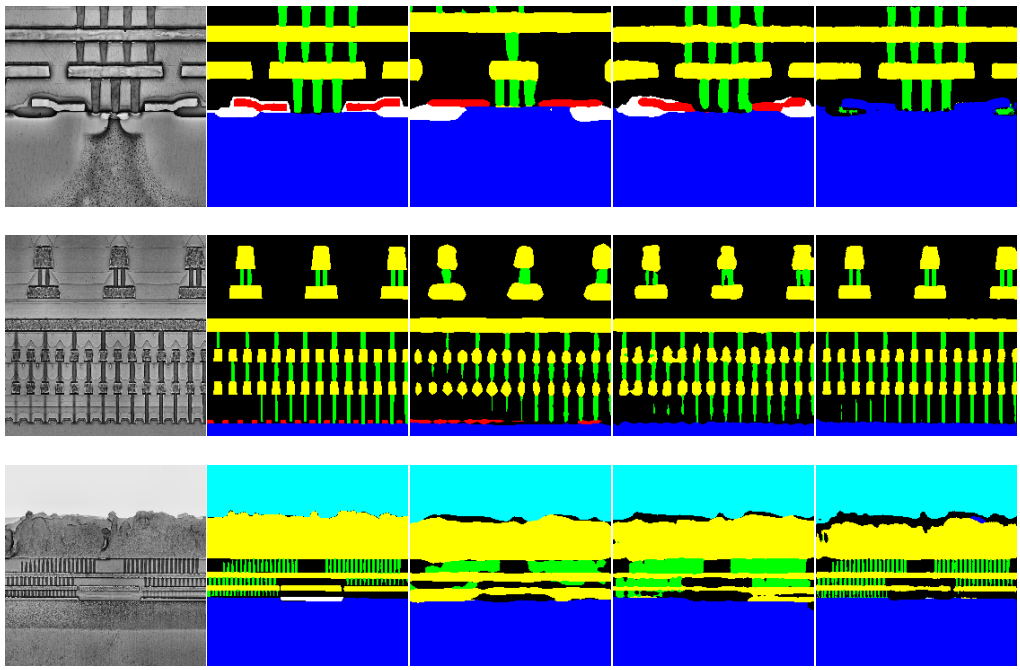


Figure 9.8: Input-target pairs and exemplary output masks of the Baseline, lightweight GSCNN (L-GSCNN) and PUBNet. The figure is based on the corresponding original publication [144].

10 Conclusion and Outlook

This thesis demonstrates the impact of domain knowledge exploitation on ML applications in the context of AD, smart infrastructure, and manufacturing, specifically in the automotive and semiconductor sectors. A common thread connects each of these topics: the use of domain expertise enables or improves the applications. Among these applications are the preparation of data, the simplification and enhancement of individual network architectures, and the design of cascaded ML systems. The work demonstrates the value of leveraging available domain knowledge to utilize limited resources efficiently. Combining knowledge modeling, ML, and deep learning techniques is key to achieving this goal. The following sections summarize the main contributions of this work, highlight the relevance for business, and outline future research directions.

10.1 Main Contributions of this Work

Knowledge-driven low-level sensor fusion

This work proposes a low-level sensor fusion approach leveraging expert domain knowledge for decreased object detection latency in chapter 4. The fusion network is designed to leverage knowledge models. Compared to state-of-the-art, the approach reduces the computational load when handling low-level sensor data. Furthermore, as physics implies, leveraging basic constraints enables data association for asynchronous sensor setups. The fusion approach has proven to be applicable for different multi-modal sensor configurations for in-vehicle and stationary perception systems based on real-world and simulated sensor data, which is demonstrated in chapter 4 and chapter 7.

Leveraging object detection data for segmentation

In chapter 5, a deep multi-modal fusion architecture for heat map-based object detection through segmentation is introduced, which uses expert domain knowledge on object appearance to leverage object data for segmentation purposes. This is achieved by fusing multi-modal sensor data early in a DNN. Segmentation masks were derived from object-level data by approximating the appearance of an object class as a heatmap based on human knowledge. The evaluation results confirm the findings of chapter 4, namely that early and low-level sensor fusion is beneficial for computational efficiency, training effort, and detection performance.

Knowledge Driven Perception Framework

A new framework for designing multi-stage perception systems is introduced in chapter 6. The framework combines knowledge engineering and NNs while reducing and addressing some of the limitations of deep learning. A key benefit is the explainability of classification traversal throughout the classification system due to the individual stages. A knowledge model on the desired use case is the foundation for identifying the individual stages of the classification system. The generality expectation of a deep learning system is replaced with expert domain knowledge. The cascaded architecture is situation driven, which allows steering classification granularity and using individual classifiers depending on environmental conditions like weather or time of day. This results in an actively manageable computational load. The modularity of the approach allows the extension of the system without introducing the need to retrain previous stages and classifiers. As exhibited in chapters 6 and 7, the proposed framework has been effectively implemented, employing real-world data from multiple multi-modal sensor systems and simulated data utilizing diverse sensor configurations.

Scalable Perception System

Chapter 7 implements object detection based on the approach introduced in chapter 4 and applies the perception framework from chapter 6 to a real-world use case to detect spectators in a large variety of environmental surroundings. It leverages a digital twin of the environment and a simulation engine to alter environmental conditions within the digital twin. This use case further demonstrates and validates the proposed approaches for knowledge exploiting training data creation, low-level sensor fusion, and knowledge-driven perception while achieving unprecedented performance for spectator classification under various environmental conditions. This use case demonstrates the feasibility of creating training data based on expert knowledge to enable the training of a perception system based on a knowledge model. This shows that deep learning and knowledge modeling approaches are complementary and enable new use cases that would otherwise not be feasible due to computational or financial constraints. The ability to simulate different sensor setups on vehicles and stationary systems allows the evaluation of software stacks before deploying them in a production system, including an estimation for computational requirements. The digital twin enables an evaluation of the knowledge model and individual classifiers, as well as a targeted and domain knowledge-driven creation of training data to achieve the required variety in data, which is the foundation for achieving the desired results.

Knowledge-Optimized Classification

The ability to reduce the complexity of NNs is one of the key contributions of this work. In chapter 8, the concept of knowledge exploitation for classification tasks with a focus on data preparation is demonstrated based on a manufacturing use case. Expert knowledge is used to collect a representative dataset, reflecting the variance one would encounter in large-scale production scenarios. Domain expertise is exploited for a classification

network design, which results in a shallow network performing comparable to a DNN, demonstrating the generalizability of the concept of knowledge exploitation. A significant achievement is the use of domain expertise for data collection and network architecture design, which translates to more efficient use of training and inference resources.

Knowledge-Driven Architecture

In chapter 9, a knowledge-driven data collection, data annotation, and segmentation process for integrated circuits is demonstrated. The proposed system architecture is unique because domain expertise is leveraged end-to-end, ranging from data collection to the resulting segmentation, combining knowledge modeling and deep learning. Known interconnections of the class-relationship graph are introduced as boundary conditions, which eliminate the need for a NN to learn this information implicitly based on training data. The NN architecture introduces elements to account for known discriminating features, further reducing the problem's complexity. The results demonstrate that the methodologies from chapter 6 to incorporate domain expertise apply to a use case found in semiconductor manufacturing. The reduced need for training data is another key enabler for this use case due to the effort required to capture and annotate training data.

10.2 Business Relevance

The approaches presented throughout this work are evaluated with real-world data, demonstrating the feasibility of applying the underlying concepts. The sensor fusion and perception, as presented in chapters 4, 6 and 7, are used at Siemens for smart infrastructure projects, like roadside units and resulted in intellectual property being covered by numerous patents [161, 162, 163, 164, 165, 166, 167, 168, 169, 170]. This proves the value of the presented methodologies for practical applications, which directly impact the business. The impact on industrial applications of the methods presented in chapters 8 and 9 is given by an automotive OEM and an IC manufactorer using the presented approaches and further enhancements or derivations for quality control of manufacturing electrical motors and the reverse engineering process of ICs.

Using ML at Scale

Industrial manufacturing at scale relies on repetitive processing steps. This introduces boundary conditions and a well-controlled environment, enabling systems to leverage ML and pattern recognition. The proposed approach of combining knowledge modeling with ML takes advantage of these constraints by explicitly modeling them, which results in more efficient and less complex systems. Training data collection is time-consuming, expensive, or both, depending on the specific process. This is caused by the need to achieve enough variance in a dataset to cover the to-be-detected classes and patterns. However, intentionally creating faulty samples or simply collecting enough samples can result in high costs, which make the application of ML prohibitively expensive for some

use cases. The data annotation is crucial for ensuring high reliability and enabling training. Many use cases require extensive domain expertise to be able to annotate data. For multi-modal labeling, a team of experts might be required to work on such a task, which results in high effort and costs. As shown throughout this work, the ability to exploit domain knowledge is a key enabler for using ML based systems at scale within the manufacturing industry while reducing system complexity.

Decreased Need for Training Data

The need for training data has constantly increased when ML based systems are intended to be used. The approaches introduced throughout this thesis demonstrate the ability to reduce the required amount of training data by leveraging domain expertise while improving performance. This is achieved by a variety of techniques, which are demonstrated in a variety of different use cases. Firstly, the constraints of an application are modeled explicitly, which reduces the required learning capacity, directly resulting in a reduced need for training data. Additionally, the data generation process is optimized based on the required variance in samples to achieve the desired results. Next, the annotation process focuses on high quality rather than quantity, which contrasts trends observed for ML applications. Reducing the impact of labeling bias introduced by labeling techniques, described in [171], reduces the required data for training. This is achieved by improving the process or approximating object appearance based on domain expertise. The reduced need for training data significantly impacts all use cases across industries where annotated data is not easily available or expensive to create. The focus on moving from a high-quantity approach to a high-quality one is gaining traction in the industry. It is generally referred to as data-centric AI, and start-ups focusing on this have evolved [172]. This leads to the suggestion that an end-to-end framework from data collection and preparation to readily usable ML applications is of great value and would be expected to be commercially successful.

Increased Computational Efficiency

The progress in computational capacity has enabled training very deep NNs. The deeper a network, the more computational resources are required during training and inference. The proposed techniques throughout this thesis improve computational efficiency in two ways. Firstly, knowledge modeling reduces the required learning capacity, which enables the usage of shallower architectures. These smaller architectures have fewer weights, reducing the computational load during training and inference. Secondly, introducing a modular perception approach, which is situation adaptive, directly results in a reduced computational load.

In contrast to a monolithic DNN, where inference complexity and classification granularity remain constant, the modular system enables situation-adaptive classification granularity and active computational resource management. The benefits of reduced computational complexity have multiple benefits for practical applications. Overall costs for training are reduced, and the required hardware at inference is also impacted. For

large-scale applications, the unit economics are improved, whereas other applications are enabled in the first place because they might not be feasible with traditional DNNs. A reduced and manageable computational load directly correlates to system latency, which is crucial for many real-world applications.

Modularity and Explainability

The proposed knowledge-driven multi-stage perception systems are built modularly, which depends on the underlying knowledge graph. During inference, the graph traversal is known and depends on the results of previous stages. The resulting benefits are twofold: This system design enables the explainability required for verifying and validating a perception system because the traversal within the classification graph is known and depends on the underlying knowledge model. This is a major advantage compared to DNNs, where the monolithic architecture prevents insights on this level. The second advantage is that modularity allows retraining individual stages of a cascaded classifier if performance improvements or additional classes are required for a specific step. Consequently, an extension with additional stages is possible. This is a key differentiator to traditional deep learning approaches, where retraining of an entire DNN would be required. This competitive advantage in industrial and large-scale applications is due to reduced effort and costs.

Decreased Latency for Fusion and Perception Systems

Object detection latency is crucial, especially in the context of safety-critical use cases. AD poses a multitude of examples where detection latency is crucial to operating an automated vehicle safely. The proposed low-level sensor fusion for object detection purposes has a major advantage compared to other detection approaches regarding latency. In contrast to systems, which fuse on the object level, the individual processing time of a subsystem based on a singular modality is avoided. Furthermore, the likelihood of detecting an object using fused sensor data is increased, resulting in earlier detections where the information of a single modality would not be sufficient to detect an object reliably.

A cascaded perception system has a reduced latency because as soon as an object relevant to the driving mission is classified, this information is available to other system components. Velocity or steering corrections can be executed before the full cascade of classifiers has been successfully executed. Individual stages of the cascade use shallow networks characterized by lower execution duration. The reduced fusion and perception latency pose stand-alone advantages, accumulating when a system combines both proposed approaches. Low detection latencies contribute to increasing the adoption rate of ML in safety-critical applications where latency requirements are crucial. This applies to AD and, more generally, to automated robotics use cases.

Digital Twin

Digital twins provide new opportunities for further increasing the adoption of ML. The ability to evaluate complex systems before implementing them in real-world environments is of great value because of reduced costs and increased likelihood of a successful project outcome. Among other important aspects, the number of sensors and their positioning, including the impact on the overall system performance, can be simulated. This allows for cost optimization of such systems regarding hardware and system integration costs while enabling the evaluation of degraded environmental conditions on system performance. The ability to evaluate the overall system performance under various conditions is valuable because system specifications can be laid out properly to meet desired requirements, or feasibility studies can be run before investing in physical systems and prototypes.

Additionally, merging real-world and simulated data can greatly influence the overall cost of generating training data. The compilation of training data for infrequent events might be prohibitively expensive but is essential to ensure the system meets the required standards or certification criteria. This prevalent industry issue can be alleviated through the strategic application of simulated data.

10.3 Future Directions of Research

This work is a starting point for directing future research that aims to leverage human domain knowledge for ML problems. The suggested methodologies and frameworks are adaptable and expandable to novel scenarios, serving as the basis for solving new challenges and sparking new ideas.

Sensor Fusion

The implications of low-level sensor fusion on object detection have been evaluated in chapters 4 and 5. In the scope of this thesis, a small subset of possible sensor modalities has been investigated while limiting the use cases that were investigated. The concept of early sensor fusion has been approached from two different angles and has proven to be beneficial. Firstly, a Bayesian network based on expert domain knowledge has been evaluated, which has to be further optimized to ensure generalizability for the desired use case. In particular, the necessity to adapt the network to changing environmental conditions needs additional research. The evaluation in chapters 6 and 7 demonstrated the feasibility of applying the approach under varying weather conditions. Still, adopting the fusion network was out of the scope of this work. Recent developments, namely transformer-based models, raise the need to investigate whether explicit modeling based on expert domain knowledge can be avoided using such an architecture [173]. If this proves to be true, the Bayesian network approach of this thesis would be seen as a workaround. Additionally, the specific implementation, as found in chapter 4, is a workaround aiming to demonstrate the potential of a low-level sensor fusion approach.

Secondly, a DNN fusing sensor data at different stages in the network has shown the best performance when fusing early within the network while leveraging domain knowledge on object appearance. This confirms the findings from the sensor fusion, which is based on a Bayesian network. The work in chapter 5 primarily focused on understanding the impact of fusion levels on detection performance. The evaluation fell short of evaluating different encoders or optimizing the network architecture further. In the future, low-level sensor fusion is set to become a key element of various applications across industries because of its superiority to higher-level fusion approaches. The combination of sensors and the choice of fusion approach will remain a key research topic for future efforts. Nevertheless, the concept of fusing at low levels generalizes to different types of data sources across applications and fusion approaches.

Leveraging Existing Annotated datasets

A prerequisite for adopting ML-based technology to different use cases is the availability of real-world sensor data. This requirement can be reduced by utilizing a digital twin and simulated data, but real-world data remains an important cost factor when implementing ML.

An approach to leveraging object detection datasets for semantic segmentation is presented in chapter 5. Object heat maps, derived from bounding box annotations, are used to approximate the appearance of objects based on human knowledge. An object detection dataset was successfully used to train a NN for semantic segmentation. This approach could be improved by combining object detection and segmentation datasets. This would entail resizing the bounding boxes for each category to a uniform size and calculating the mean of the segmentation masks located within these bounding boxes. Consequently, this would generate heat map estimates for every object class.

Investigating the possibilities to leverage existing datasets for semantic segmentation approaches has the potential to unlock tremendous potential for academia and practical applications. Further investigating how existing datasets could be repurposed to support additional use cases is an active field of research and confirms the direction of research taken in this thesis is valid while demonstrating the generalizability of the approach [174]. Further research should incorporate object orientation and how the approximated heat maps will change depending on orientation.

Knowledge Exploitation

The knowledge-driven perception framework is introduced in chapter 6 and demonstrated in chapter 7. The presented approach uses expert domain knowledge to design a cascade of classifiers instead of relying on a DNN to implicitly learn relationships to allow for generalization within the context of the use case. The results were achieved under various environmental conditions and provide evidence that the multi-stage architecture is suitable for classifying objects in an automotive use case context. Furthermore, chapters 8 and 9 show the potential of knowledge-driven NN architectures for industrial use cases.

Well-controlled manufacturing works well with the proposed knowledge exploitation as described in chapters 8 and 9. In contrast, the automotive use cases in chapters 6 and 7 revealed a potentially major disadvantage of the cascaded classification approach, which is an increased amount of individual classifiers to handle increasingly complex use cases, required to automate vehicles further. In the future, the concepts of efficiently extending and actively managing the logic during inference time required to transition across the classification hierarchy and to adapt to changing environmental conditions are subject to research. Hence, the approach of exploiting domain knowledge generalizes across various applications. At the same time, individual use cases need to be critically reviewed in relation to the effort of gathering the knowledge as a prerequisite for enablement.

Recent developments with transformer-based NN architectures could be well suited for addressing the shortcomings of the proposed approach while further generalizing and augmenting it. Knowledge-aware and topic-driven transformers are a very active field of research, as outlined in [175], [176] and [177], and could lead to much wider adoption of ML applications across different industries and academia.

Synthetic Data

This work exploits synthetic data to enable various use cases for AD and manufacturing and proves the generalizability of doing so. The simulation engine used for spectator detection in chapter 7 shows that there is a large need to further improve the quality of simulations to more closely match sensor data as expected in real-world scenarios. This ranges from enhanced physics-based sensor simulation to a more realistic approximation of human movements and natural noise caused by environmental conditions. This is an active field of research of recent enhancements in multi-modal sensor data simulation are described in [178]. Current and future research will use simulation to augment existing datasets to compensate for sparsely occurring events or situations which are too dangerous to replicate in the real world. Furthermore, digital twins of an environment are crucial to understanding the feasibility of specific use cases and identifying suitable layouts of sensor setups and even entire environments, like factories. This is a key advantage that helps guide research and development efforts while controlling the costs of future systems and their layout. Consequently, synthetic data and digital twins are of great importance to academia and industry.

The industrial applications in chapter 8 and 9 only rely on real-world data. In both cases, this is a limitation to extending the use cases further or entering similar ones in the future. Especially in the context of semiconductors, future work should consider leveraging existing layout data to simulate SEM images. This would allow one to bootstrap available data further and limit additional data collection. Similar considerations are true for more generic quality control setups in the industry, where simulation is leveraged to improve defect detection. The current research builds on this idea, and an overview is given in [179].

All investigated use cases have in common that by having more data available, future research and development will be able to enter new use cases and improve existing ones by augmenting real-world data with synthetic data from simulation engines.

Data Centric ML

A key area for future research and development is to effectively apply ML to a growing number of applications throughout different industries, where the success of implemented solutions strongly depends on the underlying data. This work provides a basis for leveraging domain knowledge to design effective system architectures and to improve training data creation and preparation by introducing generalizable approaches. A ML-based system is very tightly coupled to the data it is based on. The vision to achieve data-centricity for ML applications depends on the ability to efficiently leverage available data and knowledge to achieve an increased adoption of this technology. A solution explainable at different stages allows for targeted improvement of the overall system. Overcoming and addressing existing limitations of ML based systems by exploiting domain knowledge provides the opportunity to apply ML and AI to a large variety of new use cases. The results and concepts outlined throughout this thesis present an important milestone and starting point for other researchers to further unlock the potential of ML and AI for new use cases across academia and industry.

A Appendix

Encoder Architectures

SENet

Squeeze and Excitation Networks (SENet) are an approach for improving channel interdependency with a small increase in computing power. This method involves including a parameter for each channel, which allows the network to adjust the weighting in an adaptive manner for each feature map. Global understanding of a channel is achieved by condensing each value into a single value and pooling the global average. This results in a vector of size n, where n is the number of channels. A two-layer fully connected neural network called the excitation operator is introduced, which takes the vector as input. A vector is produced as a result of the operator; this vector has weights for each layer corresponding to the channel-wise interdependencies. The architecture is shown in figure A.1. Compared to existing architectures, SE blocks provide significant performance improvements and can be easily integrated into standard architectures such as VGG or ResNet [180].

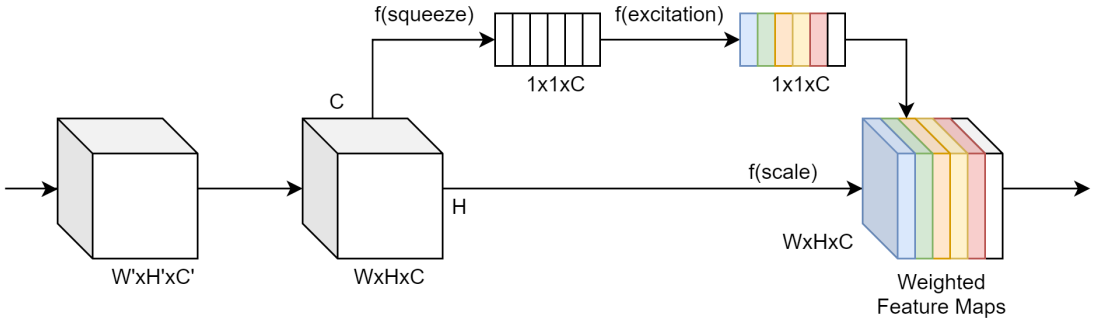


Figure A.1: Squeeze excitation block: it improves the channel interdependency by adjusting the weight of each feature map adaptively.

ResNeXt

The ResNeXt building block, based on the split-transform-merge paradigm, is similar to the Inception module of GoogleNet, which uses filters of multiple sizes that operate at the same level to create a wider rather than a deeper model [54]. In contrast to the inception block of GoogleNet, the outputs of the ResNeXt blocks are merged by adding them together. Unlike the inception block, the ResNeXt block shares the same topology

across all paths. Thus, fewer tunable parameters exist, meaning the network can be trained more easily. The cardinality parameter has been introduced, which indicates the number of independent paths in the model, allowing it to be configured with a single parameter to adjust the model capacity [181]. Figure A.2) shows a ResNeXt block.

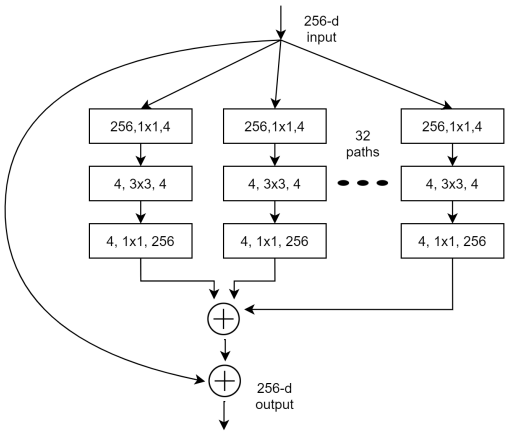


Figure A.2: A block of ResNeXt with cardinality = 32 (32 identical blocks operating at the same level). Each layer is shown as # in channels, filter size, # out channels. As in ResNet, shortcut connections are used to jump over one level.

SE-ResNet and SE-ResNeXt

It is possible to integrate squeeze excitation (SE) blocks into various standard architectures, such as ResNet or ResNeXt. As shown in figure A.3, the SE-Block operates directly on the non-identity branch. Both squeeze and excitation occur before the summation with the identity branch. Including squeeze and excitation blocks in the skip connection in ResNet-50 results in almost the same accuracy as ResNet-101. SE-Blocks, therefore, enable better results with fewer parameters [180].

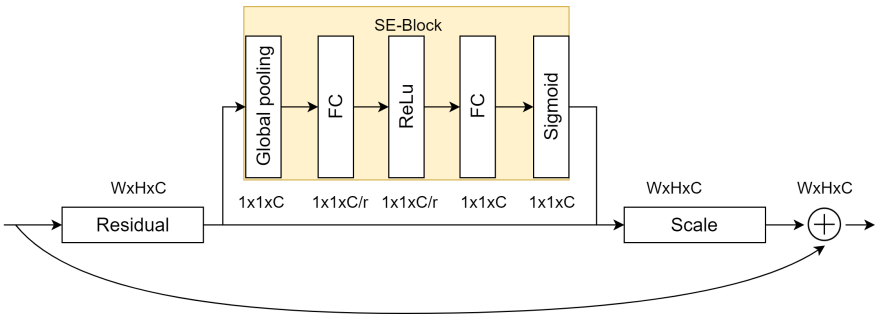


Figure A.3: SE-ResNet block. Each feature map channel is assigned a weight before it is added to the identity branch. At each layer, the dimension of the feature map is indicated. C corresponds to the channels, while r is the reduction factor of the bottleneck.

DPN

By proposing a dual path approach, the Dual Path Network (DPN) capitalizes on the excellent feature reuse properties of ResNet and combines them with exploring new feature capabilities of DenseNet. Dual path architecture allows DPN to benefit from both topologies by sharing common features while maintaining flexibility to explore new features.

Multiple modularized microblocks are stacked to form the network. Microblocks consist of a stack of 1x1, 3x3, and 1x1 convolutional layers. After the final convolutional layer, the output of the layer is split into two parts: the first part is added element-wise to the residual path, and the second part is concatenated with the densely connected component.

According to [182], DPN shows higher accuracy on PASCAL VOC 2012 tests compared to variations of ResNeXt or DenseNet.

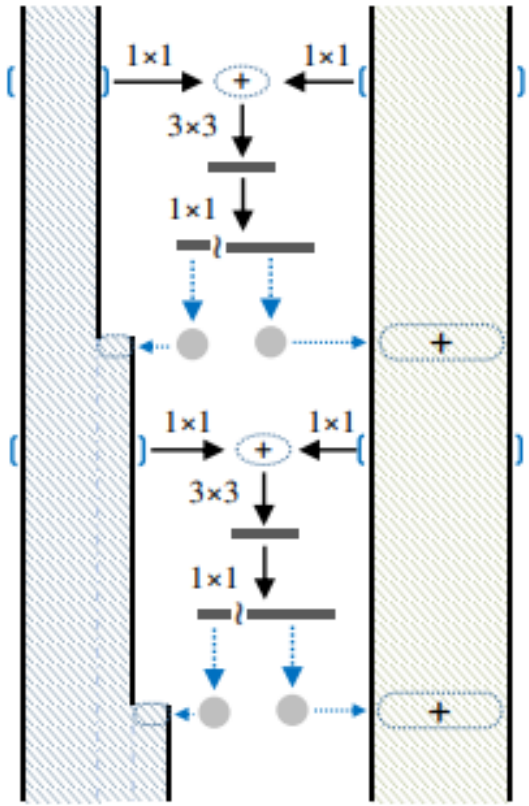


Figure A.4: Architecture of DPN, where "˜" denotes a split operation and "+" a element-wise addition.

B Acknowledgement

I would like to express my deepest gratitude to everyone who has supported me throughout my PhD journey. It is with immense pleasure and a sense of accomplishment that I acknowledge the contributions of those who have played a significant role in the completion of this thesis.

First and foremost, I would like to extend my heartfelt thanks to my professor and supervisor, Prof. Dr. Knoll, for his unwavering guidance, dedication, and encouragement. His invaluable insights, expertise, and persistent support have been instrumental in shaping my research and keeping me focused. I am truly honored to have had the opportunity to learn from and work with someone of his caliber.

I must express my profound gratitude to my loving family for their unyielding support and understanding throughout this journey. Their constant encouragement, faith in my abilities, and the occasional well-timed pep talk were the pillars that kept me going, even during the most challenging times. I dedicate this thesis to them, as a testament to their unwavering belief in me and their role in shaping the person I am today.

My heartfelt appreciation goes to Lisa, who has been my rock during this entire process. Her patience, resilience, and belief in my capabilities have been truly remarkable. She kept me on track, offered constructive criticism, and endured many late nights and moments of stress with grace and understanding. I cannot thank her enough for her love and companionship.

I would also like to extend my gratitude to my fellow PhD candidates and friends, who have shared their experiences, knowledge, and camaraderie, enriching my time as a PhD student. Their support, both academically and personally, has been invaluable and I am grateful for the friendships that have emerged from this journey.

In conclusion, I am forever grateful to everyone who has been a part of my PhD journey. The lessons I have learned, the challenges I have faced, and the relationships I have formed will remain with me for a lifetime. Thank you all for your support and belief in me.

The Use of AI

Throughout this thesis, AI was used for spelling and grammar checks, and it was employed to correct typographical or grammatical errors. Furthermore, the language clarity features of these tools foster a more concise and easy-to-follow narrative. This is geared towards enhancing readability and understandability. Additionally, AI tools were used to bolster engagement or delivery aspects and refine individual sentences to ensure smooth reading.

The use of AI tools strives to meet high standards of clarity, coherence, and precision. The tools employed for these purposes include Office365, grammarly, ChatGPT, and Wordtune [183, 184, 185, 186].

The content of the thesis is original to the author, and these tools were not leveraged for content generation. The ideas, arguments, and core content were exclusively conceived and developed by the author. As a result, the academic integrity of this thesis was preserved throughout its formulation.

List of Figures

List of Tables

Acronyms

ACC	Adaptive Cruise Control.
AD	Autonomous Driving.
ADAS	Advanced Driver-Assistance Systems.
AI	Artificial Intelligence.
ANN	Artifical Neural Network.
CAD	Computer-Aided Design.
CCD	Charge-Coupled Device.
CM	Confusion Matrix.
CNN	Convolutional Neural Network.
DNN	Deep Neural Network.
FB	Fusion Box.
FB	Fusion Box.
FCN	Fully Convolutional Network.
FMCW	Frequency-Modulated Continuous Wave.
FN	False Negative.
FOV	Field of View.
FP	False Positive.
FPN	Feature Pyramid Network.
GPS	Global Positioning System.
GSCNN	Gated Shape Convolutional Neural Network.
IC	Integrated Circuit.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
LiDAR	Light Detection and Ranging.
LLSF	Low-Level Sensor Fusion.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
NN	Neural Network.

OEM	Original Equipment Manufacturers.
PDC	Park Distance Control.
PSPNet	Pyramid Scene Parsing Network.
RADAR	Radio Detection and Ranging.
ReLU	Rectified Linear Unit.
ResNet	Residual Neural Network.
SEM	Scanning Electron Microscope.
SNR	Signal-to-Noise Ratio.
TN	True Negative.
TP	True Positive.
VGG	Visual Geometry Group.
VIA	Vertical Interconnection Accesses.

Bibliography

[1] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. doi:http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[3] Uber's safety lapses led to fatal self-driving crash, new documents suggest - The Verge. Accessed: 2023-01-30. URL: https://www.theverge.com/2019/11/6/20951385/uber-self-driving-crash-death-reason-ntsb-dcouments.

[4] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli. Trends and trajectories for explainable, accountable and intelligible systems: An HCI research agenda. *Conference on Human Factors in Computing Systems - Proceedings*, 2018-April, 4 2018. doi:10.1145/3173574.3174156.

[5] V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. 9 2019. URL: https://arxiv.org/abs/1909.03012v2.

[6] Q. V. Liao, D. Gruen, and S. Miller. Questioning the AI: Informing Design Practices for Explainable AI User Experiences. *Conference on Human Factors in Computing Systems - Proceedings*, 4 2020. URL: https://dl.acm.org/doi/10.1145/3313831.3376590, doi:10.1145/3313831.3376590.

[7] D. Kerrigan, J. Hullman, and E. Bertini. A survey of domain knowledge elicitation in applied machine learning. *Multimodal Technologies and Interaction*, 5(12):73, 12 2021. doi:10.3390/MTI5120073/S1.

[8] J. Lin, A. Zhang, M. LÃ©cuyer, J. Li, A. Panda, and S. Sen. Measuring the Effect of Training Data on Deep Learning Predictions via Randomized Experiments. *Proceedings of Machine Learning Research*, 162:13468–13504, 6 2022. URL: https://arxiv.org/abs/2206.10013v1.

[9] A. Tocchetti and M. Brambilla. The Role of Human Knowledge in Explainable AI. *Data 2022, Vol. 7, Page 93*, 7(7):93, 7 2022. doi:10.3390/DATA7070093.

[10] Road traffic injuries. Accessed: 2020-10-20. URL: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[11] Road Safety Facts — Association for Safe International Road Travel. Accessed: 2023-06-30. URL: https://www.asirt.org/safe-travel/road-safety-facts/.

[12] D. W. Martin Horn. *Automated driving.* Springer International Publishing, 1. print edition, 2017.

[13] J. Sankaran and N. Zoran. TDA2X, a SoC optimized for advanced driver assistance systems. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 2204–2208, 2014. doi:10.1109/ICASSP.2014.6853990.

[14] M. Wanitzek, M. Oehme, D. Schwarz, K. Guguieva, and J. Schulze. Ge-on-si avalanche photodiodes for LIDAR applications. *2020 43rd International Convention on Information, Communication and Electronic Technology, MIPRO 2020 - Proceedings*, pages 8–12, 9 2020. doi:10.23919/MIPRO48935.2020.9245425.

[15] H. H. Meinel. Evolving automotive radar - From the very beginnings into the future. *8th European Conference on Antennas and Propagation, EuCAP 2014*, pages 3107–3114, 2014. doi:10.1109/EUCAP.2014.6902486.

[16] K. D. Nico Kaempchen. Data synchronization strategies for multi-sensor fusion, 2003.

[17] D. Cassanelli, S. Cattini, L. Ferrari, and L. Rovati. A method for the estimate erroneous fog detection in automotive LiDAR. *2023 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 01–06, 5 2023. doi:10.1109/I2MTC53148.2023.10176011.

[18] ISO - ISO 26262-1:2011 - Road vehicles — Functional safety — Part 1: Vocabulary. Accessed: 2020-10-20. URL: https://www.iso.org/standard/43464.html.

[19] R. Schubert, C. Adam, M. Obst, N. Mattern, V. Leonhardt, and G. Wanielik. Empirical evaluation of vehicular models for ego motion estimation. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 534–539, 2011. doi:10.1109/IVS.2011.5940526.

[20] Z. Taylor and J. Nieto. Motion-Based Calibration of Multimodal Sensor Extrinsics and Timing Offset Estimation. *IEEE Transactions on Robotics*, 32(5):1215–1229, 10 2016. doi:10.1109/TRO.2016.2596771.

[21] M. Mahlisch, R. Schweiger, W. Ritter, and K. Dietmayer. Sensorfusion using spatio-temporal aligned video and lidar for improved vehicle detection. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 424–429, 2006. doi:10.1109/IVS.2006.1689665.

[22] H. C. Born, F. Oehler, V. Platte, A. Kampker, H. Heimes, B. Dorn, F. Brans, D. Drexler, F. S. L. Blanc, and S. Reising. Manufacturing Process and Design Requirements of Litz Wire with Focus on Efficiency Improvement of Traction Motors. *2022 12th International Electric Drives Production Conference, EDPC 2022 - Proceedings*, 2022. doi:10.1109/EDPC56367.2022.10019744.

[23] P. A. Schmidt and M. F. Zaeh. Laser beam welding of electrical contacts of lithium-ion batteries for electric- and hybrid-electric vehicles. *Production Engineering 2015 9:5*, 9(5):593–599, 10 2015. URL: https://link.springer.com/article/10.1007/s11740-015-0637-4, doi:10.1007/S11740-015-0637-4.

[24] H. Geng. *Semiconductor manufacturing handbook*. McGraw-Hill Education, New York and Chicago and San Francisco, second edition edition, 2018.

[25] D. Laws. 13 sextillion & counting: The long & winding road to the most frequently manufactured human artifact in history, 2018. Accessed: 2020-05-05. URL: https://computerhistory.org/blog/13-sextillion-counting-the-long-winding-road-to-the-most-frequently-manufactured-human-artifact-in-history/?key=13-sextillion-counting-the-long-winding-road-to-the-most-frequently-manufactured-human-artifact-in-history.

[26] W. Chou, L. Chen, J. Shao, A. Chen, R. Chung, and L. Zhou. Semiconductors – the next wave: Opportunities and winning strategies for semiconductor companies, 2019. Accessed: 2020-05-05. URL: https://www2.deloitte.com/cn/en/pages/technology-media-and-telecommunications/articles/semiconductors-the-next-wave-2019.html.

[27] Benjamin Benz. Vom sand zum chip: So entsteht ein moderner prozessor. *c't*, pages 76–83, 2013. Accessed: 2023-06-30. URL: https://www.heise.de/hintergrund/Vom-Sand-zum-Chip-So-entsteht-ein-moderner-Prozessor-6550454.html.

[28] OpenCV: Camera Calibration and 3D Reconstruction. Accessed: 2020-10-20. URL: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html.

[29] H. Cho, Y.-W. Seo, B. Vijaya Kumar, and R. Rajkumar. A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843, 2014. doi:10.1109/ICRA.2014.6907100.

[30] G. Daniel, W. Miao, S. Michael, and T. Ganjineh. Radar/lidar sensor fusion for car-following on highways. *5th International Conference on Automation, Robotics and Applications (ICARA)*, pages 407–412, 2011.

[31] K. C. Fuerstenberg, K. C. Dietmayer, and V. Willhoeft. Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. *IEEE Intelligent Vehicles Symposium, Proceedings*, 1:31–35, 2003. doi:10.1109/IVS.2002.1187923.

[32] K. D. Nico Kaempchen, Matthias Buehler. Feature-level fusion for free-form object tracking using laserscanner and video. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2005:453–458, 2005. doi:10.1109/IVS.2005.1505145.

[33] M. Goppelt, H.-L. Blöcher, and W. Menzel. Automotive radar-investigation of mutual interference mechanisms. *Adv. Radio Sci*, 8:55–60, 2010. URL: www.adv-radio-sci.net/8/55/2010/, doi:10.5194/ars-8-55-2010.

[34] N. Kaempchen. Feature-Level Fusion of Laser Scanner and Video Data for Advanced Driver Assistance Systems. In *Universität Ulm*, 2007.

[35] b. L. David Hall, S. A. H McMullen Ge Wang, H. L. David, and M. A. Sonya. Hall David L, McMullen Sonya AH: Mathematical Techniques in Multisensor Data Fusion. *BioMedical Engineering OnLine 2005 4:1*, 4(1):1–2, 3 2005. Accessed: 2020-10-20. URL: https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-4-23, doi:10.1186/1475-925X-4-23.

[36] J. Pearl. Probabilistic Reasoning in Intelligent Systems. *Probabilistic Reasoning in Intelligent Systems*, 1988. doi:10.1016/C2009-0-27609-4.

[37] H. A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007. doi:10.1109/JPROC.2007.896497.

[38] L. Mercep. Context-centric design of automotive human-machine interfaces, 2014.

[39] F. Jondral and A. Wiesler. Wahrscheinlichkeitsrechnung und stochastische Prozesse. *Wahrscheinlichkeitsrechnung und stochastische Prozesse*, 2002. doi:10.1007/978-3-663-01598-7.

[40] F. Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 7 1991. doi:10.1016/0925-2312(91)90023-5.

[41] K. O'Shea and R. Nash. An Introduction to Convolutional Neural Networks. *International Journal for Research in Applied Science and Engineering Technology*, 10(12):943–947, 11 2015. doi:10.22214/ijraset.2022.47789.

[42] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. Accessed: 2020-10-20. URL: https://arxiv.org/pdf/1603.07285.

[43] H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[44] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. Accessed: 2020-10-20. URL: https://arxiv.org/pdf/1311.2901.

[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[46] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[47] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi:10.1109/CVPR.2014.81.

[48] R. Girshick. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1440–1448, 2015. doi:10.1109/ICCV.2015.169.

[49] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi:10.1109/TPAMI.2016.2577031.

[50] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[51] Large scale visual recognition challenge (ilsvrc). Accessed: 2022-08-20. URL: http://www.image-net.org/challenges/LSVRC/.

[52] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi:10.1145/3065386.

[53] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1409.1556.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1409.4842.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1512.03385.

[56] K. Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. doi:10.1007/bf00344251.

[57] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 2012. doi:10.1145/3065386.

[58] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1608.06993.

[59] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/2001.05566.

[60] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1505.04597.

[61] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1612.01105.

[62] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollàr. Learning to refine object segments. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1603.08695.

[63] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1612.03144.

[64] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1901.02446.

[65] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1907.05740.

[66] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem solving program. *IFIP congress*, 256:64, 1959.

[67] Introducing the Knowledge Graph: things, not strings. Accessed: 2023-06-30. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[68] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514, 2021.

[69] Google Knowledge Graph Search API — Google Developers. Accessed: 2023-06-30. URL: https://developers.google.com/knowledge-graph.

[70] C. Shorten and T. M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):1–48, 12 2019. URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0, doi:10.1186/S40537-019-0197-0/FIGURES/33.

[71] J. R. Vargas Rivero, T. Gerbich, B. Buschardt, and J. Chen. Data Augmentation of Automotive LIDAR Point Clouds under Adverse Weather Situations. *Sensors (Basel, Switzerland)*, 21(13), 7 2021. doi:10.3390/S21134503.

[72] Q. Zhang, W. Wang, and S. C. Zhu. Examining CNN Representations With Respect to Dataset Bias. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):4464–4473, 4 2018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11833, doi:10.1609/AAAI.V32I1.11833.

[73] D. Cremers. Image segmentation with shape priors: Explicit versus implicit representations. *Handbook of Mathematical Methods in Imaging*, 2:1909–1944, 2015.

[74] M. Pollach, F. Schiegg, and A. Knoll. Low Latency and Low-Level Sensor Fusion for Automotive Use-Cases. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6780–6786, 5 2020. doi:10.1109/ICRA40945.2020.9196717.

[75] C. Tarin, H. Brugger, R. Moscardo, B. Tibken, and E. P. Hofer. Low level sensor fusion for autonomous mobile robot navigation. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 3:1377–1382, 1999. doi:10.1109/IMTC.1999.776031.

[76] S. Blank, T. Föhst, and K. Berns. A fuzzy approach to low level sensor fusion with limited system knowledge. *13th Conference on Information Fusion, Fusion 2010*, 2010. doi:10.1109/ICIF.2010.5711876.

[77] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I:886–893, 2005. doi:10.1109/CVPR.2005.177.

[78] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[79] J. Zheng, T. Su, W. Zhu, X. He, and Q. H. Liu. Radar High-Speed Target Detection Based on the Scaled Inverse Fourier Transform. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(3):1108–1119, 3 2015. doi:10.1109/JSTARS.2014.2368174.

[80] H. A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007. doi:10.1109/JPROC.2007.896497.

[81] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. doi:10.1109/ACCESS.2020.2983149.

[82] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 873–880, New York, NY, USA, 2009. Association for Computing Machinery. URL: https://doi.org/10.1145/1553374.1553486, doi:10.1145/1553374.1553486.

[83] C. D. C., M. U., G. L. M., and S. J. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010. doi:10.1162/NECO_a_00052.

[84] 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*, pages 770–778, 2016.

[85] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. Accessed: 2023-06-30. URL: https://www.bibsonomy.org/bibtex/21e476a44125e2b2b588d2fafbb5f69b0/idsia.

[86] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. doi:10.1109/ICCV.2015.123.

[87] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[88] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[89] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. doi:10.1109/CVPR.2017.243.

[90] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[91] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: the kitti dataset. *The International Journal of Robotics Research*, 32:1231–1237, 09 2013. doi:10.1177/0278364913491297.

[92] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[93] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[94] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020. doi:10.1109/CVPR42600.2020.01164.

[95] Y. Zhang, D. Sidibé, O. Morel, and F. Mériaudeau. Deep multimodal fusion for semantic image segmentation: A survey. *Image and Vision Computing*, page 104042, 2020. URL: http://www.sciencedirect.com/science/article/pii/S0262885620301748, doi:https://doi.org/10.1016/j.imavis.2020.104042.

[96] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami. Sensor modality fusion with cnns for ugv autonomous driving in indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1531–1536, 2017. doi:10.1109/IROS.2017.8205958.

[97] Y. Hou, Y. Song, X. Hao, Y. Shen, and M. Qian. Multispectral pedestrian detection based on deep convolutional neural networks. In *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–4, 2017. doi:10.1109/ICSPCC.2017.8242507.

[98] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018.

[99] Y. Zhang, O. Morel, M. Blanchon, R. Seulin, M. Rastgoo, and D. Sidibé. Exploration of Deep Learning-based Multimodal Fusion for Semantic Road Scene Segmentation. In *VISAPP 2019 14th International Conference on Computer Vision Theory and Applications*, Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Prague, Czech Republic, 2019. doi:10.5220/0007360403360343.

[100] M. Valente, C. Joly, and A. d. L. Fortelle. Deep sensor fusion for real-time odometry estimation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6679–6685, 2019. doi:10.1109/IROS40897.2019.8967803.

[101] 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*, November 2016.

[102] Q. Zhang, X. Hu, Z. Su, and Z. Song. 3d car-detection based on a mobile deep sensor fusion model and real-scene applications. *PLOS ONE*, 15(9):1–18, 09 2020. URL: https://doi.org/10.1371/journal.pone.0236947, doi:10.1371/journal.pone.0236947.

[103] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and M. Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7, 2019. doi:10.1109/SDF.2019.8916629.

[104] S. Vora, A. H. Lang, B. Helou, and O. Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4603–4611, 2020. doi:10.1109/CVPR42600.2020.00466.

[105] Y. Li, L. Ma, Z. Zhong, F. Liu, M. Chapman, D. Cao, and J. Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21, 08 2020. doi:10.1109/TNNLS.2020.3015992.

[106] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

[107] 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 D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[108] 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*, pages 3213–3223, 2016.

[109] M. Gharbi, T. M. Li, M. Aittala, J. Lehtinen, and F. Durand. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (TOG)*, 38(4), 7 2019. URL: https://dl.acm.org/doi/10.1145/3306346.3322954, doi:10.1145/3306346.3322954.

[110] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, Cambridge, Massachusetts and London, England, 2016. Accessed: 2023-06-30. URL: `http://www.deeplearningbook.org/`.

[111] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. `doi:10.1109/ICCV.2017.324`.

[112] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, 2016. `doi:10.1109/CVPR.2016.89`.

[113] A. Varischio, F. Mandruzzato, M. Bullo, M. Giordani, P. Testolina, and M. Zorzi. Hybrid Point Cloud Semantic Compression for Automotive Sensors: A Performance Evaluation. *IEEE International Conference on Communications*, 6 2021. `doi:10.1109/ICC42927.2021.9500523`.

[114] J. Hertz, A. Krogh, R. G. Palmer, and H. Horner. Introduction to the theory of neural computation. *Physics Today*, 44(12):70, 1991.

[115] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature 2015 521:7553*, 521(7553):436–444, 5 2015. URL: `https://www.nature.com/articles/nature14539`, `doi:10.1038/nature14539`.

[116] H. L. Dreyfus and S. E. Dreyfus. What artificial experts can and cannot do. *AI & SOCIETY 1992 6:1*, 6(1):18–26, 1 1992. URL: `https://link.springer.com/article/10.1007/BF02472766`, `doi:10.1007/BF02472766`.

[117] T. Wiatowski and H. Bolcskei. A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. *IEEE Transactions on Information Theory*, 64(3):1845–1866, 12 2015. URL: `https://arxiv.org/abs/1512.06293v3`, `doi:10.48550/arxiv.1512.06293`.

[118] A. B. Koku, A. Cakir, M. Parlaktuna, and A. Sekmen. To Train or not to Train. *IEEE International Conference on Control and Automation, ICCA*, 2018-June:835–840, 8 2018. `doi:10.1109/ICCA.2018.8444165`.

[119] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12):2295–2329, 12 2017. `doi:10.1109/JPROC.2017.2761740`.

[120] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2633–2642, 5 2018. URL: `https://arxiv.org/abs/1805.04687v2`, `doi:10.48550/arxiv.1805.04687`.

[121] J. M. Alvarez and M. Salzmann. Learning the Number of Neurons in Deep Networks. *Advances in Neural Information Processing Systems*, pages 2270–2278, 11 2016. URL: https://arxiv.org/abs/1611.06321v3, doi:10.48550/arxiv.1611.06321.

[122] J. S. Judd. *Neural network design and the complexity of learning*. MIT press, 1990.

[123] C. Deng, X. Ji, C. Rainey, J. Zhang, and W. Lu. Integrating Machine Learning with Human Knowledge. *iScience*, 23(11):101656, 11 2020. doi:10.1016/J.ISCI.2020.101656.

[124] Simcenter Prescan Software simulation platform — Siemens Software. Accessed: 2023-06-30. URL: https://plm.sw.siemens.com/de-DE/simcenter/autonomous-vehicle-solutions/prescan/.

[125] Das stärkste Werkzeug für 3D-Echtzeit-Entwicklung - Unreal Engine. Accessed: 2023-06-30. URL: https://www.unrealengine.com/de.

[126] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. URL: http://ieeexplore.ieee.org/document/7353481/, doi:10.1109/IROS.2015.7353481.

[127] C. R. Albrecht, D. Nevir, A.-C. Hildebrandt, S. Kraus, and U. Stilla. Investigation on misclassification of pedestrians as poles by simulation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 804–809, 2021. doi:10.1109/IV48863.2021.9575583.

[128] Vision zero - the fia has a clear vision to reduce deaths and serious injuries in motor sport to zero. Accessed: 2022-09-20. URL: https://www.fia.com/sites/default/files/auto_medical_22_final.pdf.

[129] R. U. Islam, H. Mohammad Shahadat, and K. Andersson. A novel anomaly detection algorithm for sensor data under uncertainty. *Soft Computing*, 22:1623–1639, 2018. URL: https://doi.org/10.1007/s00500-016-2425-2, doi:10.1007/s00500-016-2425-2.

[130] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021. doi:10.1109/TETCI.2021.3100641.

[131] M. Gottinger, M. Hoffmann, M. Christmann, M. Schütz, F. Kirsch, P. Gulden, and M. Vossiek. Coherent automotive radar networks: The next generation of radar-based imaging and mapping. *IEEE Journal of Microwaves*, 1(1):149–163, 2021.

[132] Open Scene Graph. Accessed: 2023-06-30. URL: http://www.openscenegraph.com/.

[133] A. Kampker, P. Burggräf, C. Deutskens, and C. Niebuhr. Competitive strategies for value creation during disruptive innovations. In *Competitive Manufacturing Conference. Stellenbosch*. Citeseer, 2013.

[134] J. Vater, M. Pollach, C. Lenz, D. Winkle, and A. Knoll. Quality control and fault classification of laser welded hairpins in electrical motors. *European Signal Processing Conference*, 2021-January:1377–1381, 1 2021. doi:10.23919/EUSIPCO47968.2020.9287701.

[135] R. Girshick. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1440–1448, 2015. doi:10.1109/ICCV.2015.169.

[136] A. Mayr, A. Meyer, J. Seefried, M. Weigelt, B. Lutz, D. Sultani, M. Hampl, and J. Franke. Potentials of machine learning in electric drives production using the example of contacting processes and selective magnet assembly. *2017 7th International Electric Drives Production Conference, EDPC 2017 - Proceedings*, 2017-December:1–8, 3 2018. doi:10.1109/EDPC.2017.8328166.

[137] M. Weigelt, A. Mayr, J. Seefried, P. Heisler, and J. Franke. Conceptual design of an intelligent ultrasonic crimping process using machine learning algorithms. *Procedia Manufacturing*, 17:78–85, 2018.

[138] A. Mayr, M. Weigelt, J. Von Lindenfels, J. Seefried, M. Ziegler, A. Mahr, N. Urban, A. Kuhl, F. Huttel, and J. Franke. Electric Motor Production 4.0 - Application Potentials of Industry 4.0 Technologies in the Manufacturing of Electric Motors. *2018 8th International Electric Drives Production Conference, EDPC 2018 - Proceedings*, 3 2019. doi:10.1109/EDPC.2018.8658294.

[139] A. Mayr, B. Lutz, M. Weigelt, T. Glabel, D. Kibkalt, M. Masuch, A. Riedel, and J. Franke. Evaluation of Machine Learning for Quality Monitoring of Laser Welding Using the Example of the Contacting of Hairpin Windings. *2018 8th International Electric Drives Production Conference, EDPC 2018 - Proceedings*, 3 2019. doi:10.1109/EDPC.2018.8658346.

[140] A. Riedel, M. Masuch, M. Weigelt, T. Glabel, A. Kuhl, S. Reinstein, and J. Franke. Challenges of the Hairpin Technology for Production Techniques. *ICEMS 2018 - 2018 21st International Conference on Electrical Machines and Systems*, pages 2471–2476, 11 2018. doi:10.23919/ICEMS.2018.8549105.

[141] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 5 2016. URL: https://pubmed.ncbi.nlm.nih.gov/26886976/, doi:10.1109/TMI.2016.2528162.

[142] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[143] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[144] M. Pollach, F. Schiegg, M. Ludwig, A. C. Bette, and A. Knoll. Boundary Enhanced Semantic Segmentation for High Resolution Electron Microscope Images. *European Signal Processing Conference*, 2022-August:523–527, 2022. doi:10.23919/EUSIPCO55093.2022.9909919.

[145] U. Guin, D. Dimase, and M. Tehranipoor. Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead. *Journal of Electronic Testing: Theory and Applications*, 30:9–23, 02 2014. doi:10.1007/s10836-013-5430-8.

[146] R. Elnaggar and K. Chakrabarty. Machine learning for hardware security: opportunities and risks. *Journal of Electronic Testing*, 34(2):183–201, 2018.

[147] U. J. Botero, R. Wilson, H. Lu, M. T. Rahman, M. A. Mallaiyan, F. Ganji, N. Asadizanjani, M. M. Tehranipoor, D. L. Woodard, and D. Forte. Hardware trust and assurance through reverse engineering: A survey and outlook from image analysis and machine learning perspectives. *arXiv preprint arXiv:2002.04210*, 2020.

[148] X. Hong, D. Cheng, Y. Shi, T. Lin, and B. H. Gwee. Deep learning for automatic ic image analysis. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2018.

[149] D. Cheng, Y. Shi, T. Lin, B.-H. Gwee, and K.-A. Toh. Hybrid k-means clustering and support vector machine method for via and metal line detections in delayered ic images. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(12):1849–1853, 2018.

[150] N. Asadizanjani, M. Tehranipoor, and D. Forte. Counterfeit electronics detection using image processing and machine learning. *Journal of Physics: Conference Series*, 787:012023, jan 2017. doi:10.1088/1742-6596/787/1/012023.

[151] P. Ghosh and R. S. Chakraborty. Recycled and remarked counterfeit integrated circuit detection by image-processing-based package texture and indent analysis. *IEEE Transactions on Industrial Informatics*, 15(4):1966–1974, 2019.

[152] R. Hammond. Counterfeit electronic component detection. Accessed: 2020-10-20. URL: https://www.aeri.com/counterfeit-electronic-component-detection/.

[153] V. Iglovikov and A. Shvets. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1801.05746.

[154] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. Accessed: 2023-06-30. URL: `https://arxiv.org/pdf/1409.0575`.

[155] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. Accessed: 2023-06-30. URL: `https://arxiv.org/pdf/1802.02611`.

[156] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, 2014. `doi:10.1109/CVPR.2014.222`.

[157] A. Schindler, T. Lidy, and A. Rauber. Comparing shallow versus deep neural network architectures for automatic music genre classification. *Proceedings of the 9th Forum Media Technology 2016*, 1734, 2016. Accessed: 2020-05-05. URL: `https://publik.tuwien.ac.at/files/publik_256008.pdf`.

[158] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. Accessed: 2023-06-30. URL: `https://arxiv.org/pdf/1706.02677`.

[159] D. Masters and C. Luschi. Revisiting small batch training for deep neural networks. Accessed: 2023-06-30. URL: `https://arxiv.org/pdf/1804.07612`.

[160] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. Accessed: 2023-06-30. URL: `https://arxiv.org/pdf/1412.6980`.

[161] L. Mercep and M. Pollach. Event classification and object tracking, US 10520904 B2, 2019-12-31, Granted Patent, United States.

[162] R. Oder, A. E. Geiger, L. Mercep, and M. Pollach. Event-driven region of interest management, US 11067996 B2, 2021-07-20, Granted Patent, United States.

[163] R. Oder, A. E. Geiger, L. Mercep, and M. Pollach. Low-level sensor fusion, US 10317901 B2, 2019-06-11, Granted Patent, United States.

[164] L. Mercep and M. Pollach. Object recognition and classification using multiple sensor modalities, US 10740658 B2, 2020-08-11, Granted Patent, United States.

[165] L. Mercep and M. Pollach. Map building with sensor measurements, US 10558185 B2, 2020-02-11, Granted Patent, United States.

[166] L. Mercep and M. Pollach. Self-diagnosis of faults in an autonomous driving system, US 11145146 B2, 2021-10-12, Granted Patent, United States.

[167] L. Mercep and M. Pollach. Sensor event detection and fusion, US 10802450 B2, 2020-10-13, Granted Patent, United States.

[168] L. Mercep and M. Pollach. Sensor modification based on an annotated environmental model, US 10678240 B2, 2020-06-09, Granted Patent, United States.

[169] L. Mercep and M. Pollach. Training of machine learning sensor data classification system, US 10884409 B2, 2021-01-05, Granted Patent, United States.

[170] L. Mercep and M. Pollach. Vehicle localization with map-matched sensor measurements, US 10585409 B2, 2020-03-10, Granted Patent, United States.

[171] C. Sager, C. Janiesch, and P. Zschech. A survey of image labelling for computer vision applications. *Journal of Business Analytics*, 4(2):91–110, 2021. doi:10.1080/2573234X.2021.1908861.

[172] Home - Landing AI. Accessed: 2023-01-30. URL: https://landing.ai/.

[173] Transformers and explainable sensor-fusion are the key technologies behind the Mapless Autonomy Platform. Accessed: 2023-07-30. URL: https://driveblocks.ai/news/2023-07-03/transformers-and-explainable-sensor-fusion-are-the-key-technologies-behind-the-mapless-autonomy-platform.

[174] H. Kervadec, J. Dolz, S. Wang, E. Granger, and I. B. Ayed. Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision. In *Medical imaging with deep learning*, pages 365–381. PMLR, 2020.

[175] L. Zhu, G. Pergola, L. Gui, D. Zhou, and Y. He. Topic-driven and knowledge-aware transformer for dialogue emotion detection. *arXiv preprint arXiv:2106.01071*, 2021.

[176] R. Li, Z. Jiang, L. Wang, X. Lu, M. Zhao, and D. Chen. Enhancing Transformer-based language models with commonsense representations for knowledge-driven machine comprehension. *Knowledge-Based Systems*, 220:106936, 5 2021. doi:10.1016/J.KNOSYS.2021.106936.

[177] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[178] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426. IEEE, 2022.

[179] D. Gauder, J. Gölz, N. Jung, and G. Lanza. Development of an adaptive quality control loop in micro-production using machine learning, analytical gear simulation, and inline focus variation metrology for zero defect manufacturing. *Computers in Industry*, 144:103799, 1 2023. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166361522001956, doi:10.1016/J.COMPIND.2022.103799.

[180] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1709.01507.

[181] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1611.05431.

[182] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. Accessed: 2023-06-30. URL: https://arxiv.org/pdf/1707.01629.

[183] ChatGPT. Accessed: 2023-06-30. URL: https://openai.com/chatgpt.

[184] Product Features — Grammarly. Accessed: 2023-06-30. URL: https://www.grammarly.com/features.

[185] Wordtune — Your personal writing assistant & editor. Accessed: 2023-06-30. URL: https://www.wordtune.com/.

[186] Microsoft 365 – Abonnement für Office-Anwendungen — Microsoft 365. Accessed: 2023-06-30. URL: https://www.microsoft.com/de-de/microsoft-365.