Neuromorphic Perception using Time-of-Flight-based Encoding of Lidar Data

A Potential Analysis and Feasibility Study

Masterthesis No. 874/23
Editor: Jonas Valentin Schulte B.Sc.
Supervisor: Lorenz Bayerlain M.Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

FZD FAHRZEUGTECHNIK TU DARMSTADT

Jonas Valentin Schulte B.Sc.
Study program: Maschinenbau

Masterthesis No. 874/23
Topic: Neuromorphic Perception using Time-of-Flight-based Encoding of Lidar Data

Submitted: November 13, 2023
Revised: February 9, 2024

Technische Universität Darmstadt
Fachgebiet Fahrzeugtechnik
Prof. Dr.-Ing. Steven Peters
Otto-Berndt-Straße 2
64287 Darmstadt

Abstract

This master thesis is dedicated to the research of neuromorphic perception in the domain of automated driving systems. In view of the rapid progress in the automotive industry and the increasing realization of automated driving, questions regarding the energy efficiency and performance of such systems are coming into focus. This work discusses the possibility of processing lidar data using spiking neural networks to reduce energy consumption in object detection while increasing perceptual capabilities. Automated vehicles not only promise a radical change in the transport sector, but also numerous benefits beyond pure mobility. These include reducing road accidents, reducing congestion, improving access to mobility and saving time in traffic. However, the implementation of automated driving systems faces technical and legal challenges, including the allocation of liability in the event of an accident and the need for intensive research into technical implementation. A key challenge is the perception of the vehicle, which is essential in order to maneuver safely and react to changing traffic conditions. Automated vehicles rely on various sensor systems and often use artificial intelligence methods to recognize objects and perform other tasks for Automated driving. However, these methods require considerable computing power and consume a lot of energy, which is particularly challenging in the context of electromobility, where energy is often limited by battery capacity. This thesis investigates a promising solution to this problem by integrating neuromorphic technology into the perception system. This technology is inspired by the neurobiological information processing of the human brain and aims to develop hardware and software that are similar in function and structure to biological neurons. Because the brain uses discrete impulses to process information, it consumes very little energy compared to conventional computers for comparable tasks. Energy is only consumed when these pulses occur and information is processed. This work focuses on the use of lidar sensors, which emit infrared light and measure the reflected laser beams to determine the distance to objects in the environment. An important parameter is the time of flight of the laser beams. The ToF can be used to convert the lidar data into discrete pulses that the neuromorphic perception system communicates with. The integration of neuromorphic perception systems into automated vehicles has the potential to significantly increase energy efficiency and improve the ability to perceive the environment. In order to analyze the potential and feasibility of a neuromorphic perception system using the ToF of lidar data, a morphological analysis is performed. This analysis takes into account different approaches and aspects that have been elaborated in an extensive literature review. The perception pipeline is divided into subsystems and subfunctions, leading to two concepts that fulfill different requirements for a neuromorphic perception system. These two concepts serve as a starting point for further research and development of this promising technology. The thesis is divided into several chapters that provide a comprehensive overview of the topic of neuromorphic perception systems for automated driving. It covers the basics of automated driving systems, artificial neural networks and spiking neural networks. Lidar sensors, different coding schemes, point cloud object detection, spiking neuron models, SNN hardware implementations and existing approaches are also discussed. The results of the literature review are evaluated based on a morphological analysis. A baseline and an advanced concept of a neuromorphic perception system are presented. In conclusion, this thesis shows the enormous potential of neuromorphic perception systems for automated driving and provides an outlook on possible future developments and research directions. Reducing energy consumption is crucial to achieving the vision of efficient, sustainable and safe mobility. Further research and development in the field of neuromorphic perception technology could be one way to achieve this.

Contents

List of Symbols and Indices

Latin formula symbols:

Symbol	Unit	Description
a	$-$	Amplitude
A	Hz	Population Activity
b	$-$	Bias
c	$\mathrm{m/s}$	Speed of Light
C	F	Capacitor
d	m	Distance
E	V	Equilibrium Potential
f	Hz	Frequency
G	S	Conductivity
I	A	Current
N	$-$	Number
p	Hz	Spike Density
P	$-$	Probability Distribution
q	$-$	Membrane Recovery
R	Ω	Resistor
t	s	Time
u	V	Membrane Potential
v	Hz	Mean Firing Rate
w	$-$	Synaptic Weight
x	$-$	Input
y	$-$	Output

Greek formula symbols:

Symbol	Unit	Description
σ	$-$	Binary Neuron Output
τ	s	Time Constant
γ	$1/\mathrm{s}^2$	Frequency Slope
δ	$-$	Learning Rate
Δ	$-$	Difference
ϵ	$-$	Synaptic Kernel
η	$-$	Refractory Kernel
θ	$-$	Binary Neuron Threshold
ϑ	V	Spiking Neuron Threshold

Indices:

Symbol	Description
C	Capacitive
h	Horizontal
ion	Ion
ISI	Inter Spike Interval
k	Ion Channel
K	Potassium Ion
L	Leakage
max	Maximum
mem	Membrane
Na	Sodium Ion
$neuron$	Neuron
R	Resistive
$rest$	Resting
run	Run
v	Vertical

List of Abbreviations

ADS	Automated Driving System
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DCNN	Deep Convolutional Neural Network
DNN	Deep Neural Network
EPSP	Excitatory Postsynaptic Potential
FLOPS	Floating Point Operations
FMWC	Frequency Modulated Continuous Wave
FN	False Negative
FoV	Field of View
FP	False Positive
FPGA	Field-Programmable Gate Array
FPS	Frames per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
IEC	International Electrotechnical Commission
IoU	Intersection over Union
IPSP	Inhibitory Postsynaptic Potential
ISI	Inter-Spike-Interval
Lidar	Light Detection and Ranging
LTD	Long-Term Depression
LTP	Long-Term Potentiation
mAP	Mean Average Precision
MEMS	Micro-Electro-Mechanical Systems
NDS	nuScenes Detection Score
NIR	Near-Infrared
NMS	Non-Maximum Suppression
ODD	Operational Design Domain

OPA	Optical Phased Array
PSP	Postsynaptic Potential
PSTM	Peri-Stimulus Time Histogram
RBF	Radial Basis Functions
R-CNN	Region-based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROC	Rank-Order Coding
ROI	Region of Interest
SAE	Society of Automotive Engineers
SLAM	Simultaneous Localization and Mapping
SNN	Spiking Neural Network
SPDA	Single Photon Detector Array
SRM	Spike Response Model
STBP	Spatio-Temporal Backpropagation
STDP	Spike Timing Dependent Plasticity
SWIR	Short-Wave Infrared
TE	Total Error
ToF	Time of Flight
TP	True Positive
TTFS	Time-to-First-Spike
YOLO	You Only Look Once

List of Figures

1 Introduction

Recent advancements in the automotive industry have introduced a revolution that will transform the transportation sector as we know it. The technology of automated driving, once only seen in fiction, is now increasingly a reality. This disruptive progress not only guarantees a radical change within the transportation sector but also offers numerous advantages and possibilities beyond pure mobility. The concept of automated driving encompasses several benefits, such as reducing road accidents, reducing congestion, improving access to mobility and saving time in traffic.[1] For example almost 90% of car accidents in the US are coursed by human error.[2] This number could be significantly reduced if automated driving become reality. Moreover, its potential in enabling more efficient driving methods combined with the shift to electric powertrains is expected to significantly diminish the environmental impact of road transportation.[3] In short, automated driving promises a future of mobility that is safer, more efficient and more sustainable.

But there are many remaining technical and legal challenges to overcome in order to realize the vision of automated driving. For example in many countries, highly automated vehicles are not yet permitted on the roads due to the uncertain allocation of liability in the event of an accident.[4] Moreover, there is still a requirement for extensive research in the technical implementation of automated driving prior to the deployment of fully automated vehicles on public roads. One of them is to enable the vehicle to detect its environment and localize its position what is referred to as perception. The gathered information about the environment is fundamental to safely perform driving maneuvers and react to changing road conditions. The perception system has to detect various objects in order to avoid collisions with other traffic participants and to rule out potential damage to pedestrians or cyclist. Due to this, it is commonly accepted that automated vehicles need to rely on various sensor systems in order to be able to cope with all traffic situations and weather conditions.[5] Based on the sensor technology, methods of artificial intelligence are used to carry out object detection and many other tasks that are needed for automated driving. On the one hand, this approach promises deep insight into the description of the environment, its objects and also their potential behavior. On the other hand, these methods require an enormous amount of computing power resulting in a significant power consumption. Particularly in the field of electromobility, where the amount of energy is largely limited by the capacity of the battery, it is important to massively reduce this power consumption in order to increase the range of automated electric vehicles and reduce the environmental impact. At the same time, the profound information provided by artificial intelligence methods must not be compromised.

Therefore, this master thesis explores a promising solution to this problem by applying neuromorphic technology to the perception system. This technology is inspired by the neurobiological information processing of the human brain. The aim is to develop hardware and software that is similar in function and structure to biological neurons in the brain, in order to create more powerful and efficient artificial intelligence (AI) systems. This is plausible as the brain consumes only a small amount of the energy that

[1] Ahmed, H. U. et al.: Technology developments and impacts of connected and autonomous vehicles: An overview (2022).

[2] National Highway Traffic Safety Administration and others: National motor vehicle crash causation survey (2008).

[3] Bierstedt, J. et al.: Effects of next-generation vehicles on travel demand and highway capacity (2014).

[4] Singler, P.: Haftungsprobleme bei autonomen Fahrzeugen (2017).

[5] Wang, Y. et al.: Multi-modal 3d object detection in autonomous driving: a survey (2023).

a computer would require for similar processing tasks. This is mainly because the brain uses discrete pulses rather than floating point values to represent information. As a result, energy is consumed when spikes occur and information is processed. As previous mentioned, perception systems commonly rely on a multi-sensor layout. This thesis focuses on the use of lidar sensors. These sensors emit infrared light and detect the reflected laser beams to measure the distance to objects in the environment. An important parameter is the time taken for the laser beams to travel, which is referred to as the time of flight (ToF). The ToF can be used to encode the laser data into discrete spikes, with which the neuromorphic perceptions system communicates with. The integration of neuromorphic perception systems into self-driving vehicles has the potential to significantly increase their energy efficiency and improve their ability to react to the environment. The use of this technology could revolutionize not only automated vehicles, but also the entire field of artificial intelligence in unprecedented ways. However, it is still a matter of research how such a neuromorphic perception system can be structured and what potential can be expected with regard to the feasibility of such a system.

In order to analyze the potential and feasibility of a neuromorphic perception system using the ToF of the lidar sensor, a morphological analysis is carried out that considers different approaches and aspects that are previously elaborated in an extensive literature review. This method is based primarily on a morphological box that makes it possible to develop concepts for neuromorphic perception by combining various partial solutions. For this purpose, in the course of the thesis, the perception pipeline is divided into subsystems and subfunctions, resulting in two concepts that satisfy different requirements that a neuromorphic perception system must fulfill. The two concepts which are the result of this master thesis shall give a first assessment of the potential and feasibility of neuromorphic perception and serve as a starting point for further research and development of this promising technology.

The opening chapter introduces the background of the topic. It starts with a general introduction to automated driving systems and their system architecture, followed by a detailed consideration of the perception system. Furthermore, artificial neural networks (ANNs) are explained, starting with the perceptron and early neural networks and then delving into deep learning and convolutional neural networks. In addition, spiking neural networks (SNNs) are considered, including an explanation of biological neurons, the spike response model and the basics of data encoding. The next chapter deals with the current state of literature and research related to the topic of the thesis. It examines lidar sensors, different coding schemes, object detection with point clouds, including benchmark datasets and different projection and volume methods. In addition, spiking neuron models such as the Hodgkin-Huxley model, the integrate-and-fire model and the Izhikevich model are highlighted. The different learning rules for SNNs are also covered, including unsupervised learning, supervised learning and ANN-to-SNN conversion. Finally, the hardware implementation of SNNs and existing approaches are discussed. The fourth chapter of the thesis contains the definition of requirements for a neuromorphic perception system that uses the ToF of the lidar data and the presentation of the method used to develop two concepts of a neuromorphic perception pipeline. This is followed by the chapter that presents and discusses the results of the work. This includes an investigation of a baseline concept and a advanced concept which follow different aspects and fulfill different requirements that were discussed in the previous chapter. The final chapter summarizes the most important findings of the thesis and gives an outlook on possible future developments and research directions. Through this structured approach, the master thesis will provide a comprehensive insight into the topic of neuromorphic perception systems for automated driving.

2 Background

This chapter provides a comprehensive overview of the background information required to understand the following chapters. It begins with an introduction to automated driving systems (ADS), also known as self-driving or driverless car systems, which are technologies that enable vehicles to operate and navigate without direct human input. These systems use a combination of sensors such as cameras, radar, lidar, GPS and on-board computers to sense the environment, interpret data, make decisions and control the vehicle. We then look at artificial neural networks (ANNs), which are a central component of many ADSs and have the ability to learn from large datasets and recognize complex patterns. By training with real or simulated driving data, they can continuously improve their performance and adapt to new traffic situations. Therefore, they play an essential role in the development and progress of automated driving by enabling accurate and robust perception and decision-making. Next, we consider the concept of spiking neural networks (SNNs). Unlike conventional ANNs, SNNs mimic the behavior of biological neurons by representing information as a series of discrete spikes. This unique approach promises advantages in terms of energy efficiency and biological plausibility, making SNNs an increasingly popular area of research.

2.1 Automated Driving Systems

To categorize the capabilities of automated driving systems, the Society of Automotive Engineers (SAE) defines a classification system that assigns six different levels of driving automation to vehicles based on the amount of human intervention and driver attention required. These levels range from level 0, where the driver has the full responsibility for the vehicle, to level 5, where the vehicle performs all driving tasks in all conditions.[6] From level 3 and higher, the system has primary control of the driving task, with the driver providing secondary assistance. This is where automated driving begins. Vehicles with level 3 ADS can only operate in limited operational design domains (ODDs), such as on motorways. The driver can engage in other activities, but must always be able to regain control of the vehicle within a short time in the event of an emergency. Level 4 and 5 systems do not require human intervention. However, a level 4 ADS can only operate within a specific ODD, which can be implemented through geofencing. This level can also include the automatic termination of the driving mission by parking. As mentioned above, a level 5 vehicle can perform the full driving task in all areas and all weather conditions.[7] An overview of the six levels of driving automation is given in figure 2-1.

If a car manufacturer wants to officially introduce a level 3 vehicle on public roads, it must accept responsibility for accidents caused by a failure of the ADS.[8] This in turn means that if the manufacturer is not prepared to accept this responsibility, the vehicle will be classified as level 2, regardless of whether it is technically capable of achieving level 3.[9] This shows that in addition to the major technical challenges of automated driving, the legal framework also plays a crucial role.

[6] SAE international: Taxonomy and definitions for terms related to driving automation systems (2018).

[7] Yurtsever, E. et al.: A survey of autonomous driving: Common practices and emerging technologies (2020).

[8] Singler, P.: Haftungsprobleme bei autonomen Fahrzeugen (2017).

[9] UN Regulation: No.157 Automated Lane Keeping Systems (ALKS) (2021).

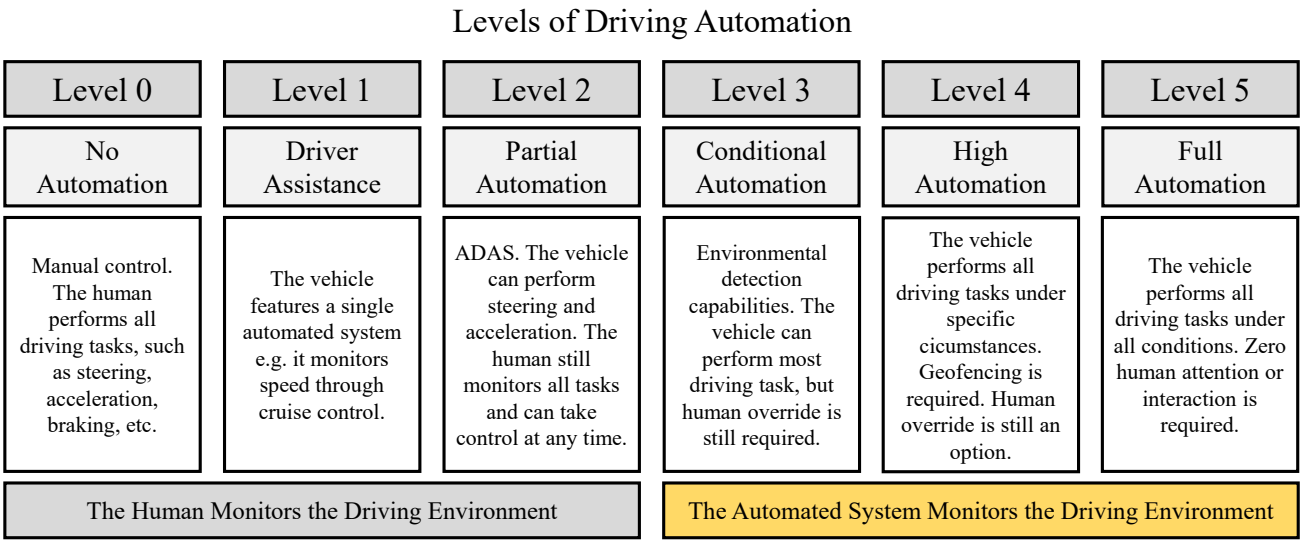

Figure 2-1: The six different levels of driving automation according to the SAE.[10]

In 2017, Audi presented their new version of the Audi A8, a vehicle that was technically capable of taking over the driving task in motorway traffic jams at speeds of up to 60 km/h.[11] However, Audi did not receive Level 3 approval for its ADS. The company has therefore abandoned its automated driving plans for the current Audi A8 model.[12] The first car manufacturer in the world to introduce an ADS with Level 3 was Mercedes in 2023. Their so-called Drive Pilot can take over the driving task in certain ODDs. For legal reasons, this system can currently only be used in Germany and the US state of Nevada.[13] In addition to series vehicles, there are also efforts to realize self-driving taxis, also known as robo-taxis. Waymo, a subsidiary of Alphabet, has developed one of the most advanced automated driving systems. Their self-driving vehicles are able to drive on public roads without human intervention. Waymo's self-driving taxis are already in use, transporting passengers in cities such as Phoenix, Arizona. Their vehicles meet Level 4 requirements and they are working on Level 5 vehicles.[14] Cruise, a company backed by General Motors, has also developed automated vehicles. Their automated vehicle, the Cruise Origin, has been designed specifically for use as an automated driving service vehicle. Cruise has plans to offer the Cruise Origin in various cities as a ride-sharing service where passengers do not need a driver. Cruise is focusing on level 4 automated driving and is testing this technology in urban environments such as San Francisco.[15] Following a serious traffic accident involving a pedestrian, the city of San Francisco revoked Cruise's driving approval for its self-driving cars in October 2023. As a result, the company initially stopped building its automated vehicles.[16] These examples show that various companies have already made great progress in the development of automated vehicles, but that many technical and legal challenges still need to be overcome.

[10] SAE international: Taxonomy and definitions for terms related to driving automation systems (2018)

[11] Ross, P. E.: The Audi A8: the World's First Production Car to Achieve Level 3 Autonomy (2017).

[12] Conrad, B.: Audi stoppt Pläne für teilautonomen A8 (2020).

[13] Hebermehl, G.; Baumann, U.: Mercedes mit Level-3-Zulassung (2023).

[14] Waymo LLC: Seeing the road ahead (2023).

[15] Cruise LLC: Driving cities forward (2023).

[16] Pape, L.: Kalifornien stoppt fahrerlose Fahrten mit Cruise-Robotaxis (2023).

2.1.1 System Architecture

At the high level of the system architecture, ADSs can be distinguished by their connectivity and algorithmic design. Accordingly, a system can be designed either as a stand-alone ego-only system or as a connected multi-agent system. An ego-only ADS performs all necessary automated driving tasks without additional information or assistance from other road users or supporting infrastructure. This approach is much more common among modern ADSs than the multi-agent design. The algorithmic design is differentiated between a modular system approach and an end-to-end driving algorithm, as shown in figure 2-2. In modular ADSs, core functions are grouped into subsystems.[17] This has the advantage that these subsystems can be developed independently, thereby breaking down the extensive task of automated driving into smaller ones.[18] However, modular systems have a tendency to propagate errors and become excessively complex.[19] A modular ADS typically comprises a perception system and a decision-making system, both of which are subdivided into several subsystems. For example, the perception system usually consists of various modules for localization, static obstacle mapping, road mapping, moving obstacle detection and tracking, and traffic signaling, among others. The decision-making system is also commonly divided into many subsystems responsible for tasks such as route planning, path planning, behavior selection, motion planning, obstacle avoidance and control. This classification of the subsystems may be slightly different depending on the ADS.[20] In the following, we discuss the perception system in more detail and outline the functions and tasks of the various subsystems.

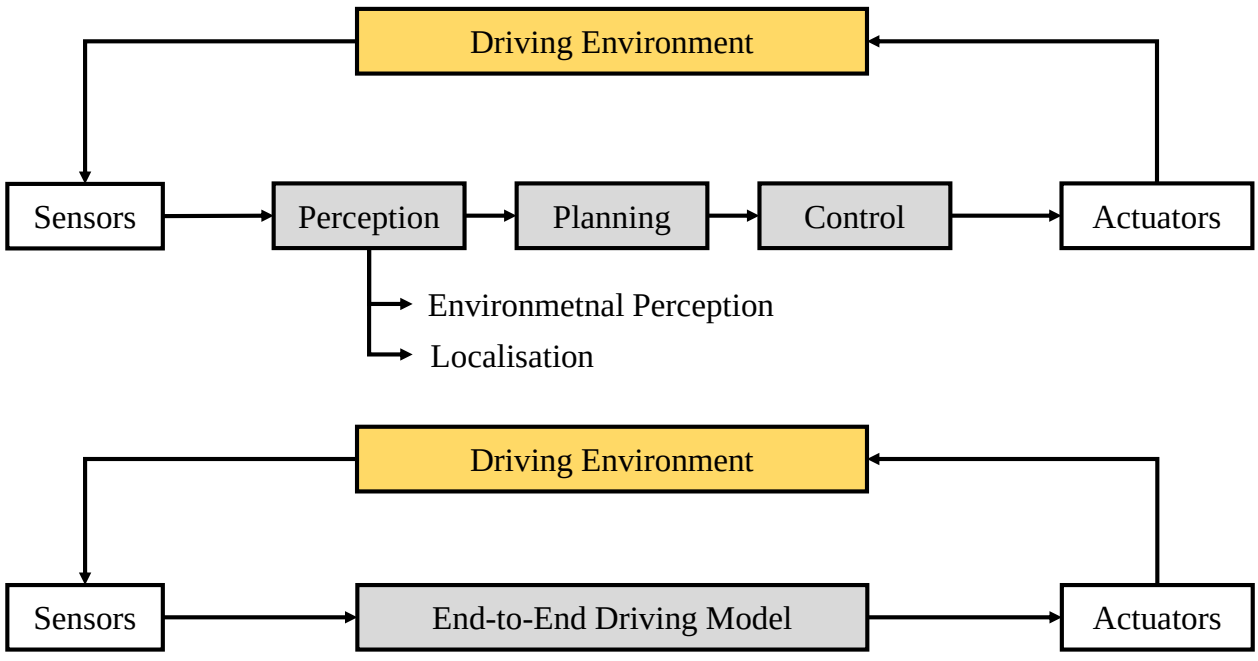

Figure 2-2: Information flow diagram of a modular ADS and an end-to-end ADS.[17]

[17] Yurtsever, E. et al.: A survey of autonomous driving: Common practices and emerging technologies (2020)

[18] Chi, L.; Mu, Y.: Deep steering: Learning end-to-end driving model from spatial and temporal visual cues (2017).

[19] McAllister, R. et al.: Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning (2017).

[20] Paden, B. et al.: A survey of motion planning and control techniques for self-driving urban vehicles (2016).

2.1.2 Perception System

The perception system supplies the ADS with a representation of the environment around the vehicle. The output of the system can be divided into three levels of information. The physical description of the environment through the pose, speed and shape of objects. The semantic description, which includes the category of objects and the intention prediction, which estimates the likelihood of an object's behavior.[21] In addition to sensor data collected from various sources such as lidar, radar, cameras, GPS, inertial measurement units, and odometers, the perception system also relies on pre-existing knowledge of sensor models, road network, traffic regulations, and vehicle behavior.[22]

In order to guide the vehicle successfully, it is essential to know exactly where the vehicle is in relation to its surroundings. This essential task is performed by the localiser, which estimates the vehicle's position and orientation relative to a map or road. Using GPS as a subsystem for localization is the simplest approach, however GPS signal can be unreliable especially in urban areas or tunnels. For this reason, many ADSs do not rely exclusively on GPS data, but use lidar, cameras or a combination of both. The lidar and camera approach aims to exploit the advantages of both sensor types.[22] Cameras are less expensive than lidars, but like humans, they struggle to perceive the environment correctly in low light. In addition, measuring the distance to other objects with camera systems is challenging. Lidar sensors, on the other hand, have high measurement accuracy in all light conditions, but struggle with fog and rain.[23,24] Many lidar- plus camera-based localisers use lidar data to generate the map and cameras to estimate the position relative to the map of the lidar sensor.[22] In this way, the localization and mapping subsystems are combined into a single system. This simultaneous localization and mapping (SLAM) approach requires no prior information about the environment and is widely used in indoor robotics. Due to the high computational requirements in outdoor conditions, SLAM algorithms are less efficient for ADSs than localization with pre-build maps.[25] The disadvantage of such map-based methods is that they are sensitive to changes in the environment. Especially in rural areas, this can lead to differences between the map and the actual environment due to roadside vegetation and structures.[26]

Another important task of the perception system is the detection and tracking of objects. Object detection usually involves identifying objects and their category in the given sensor data, and determining their size. Due to decades of research in the field of computer vision, object detection with cameras is widely used. However, object detection for complex scenarios, and thus for automated driving, only became possible through the use of deep convolutional neural networks (DCNNs).[23] A milestone in this research is the DCNN AlexNet[27], which achieved an outstanding result in the image recognition challenge ImageNet[28] in 2012.

[21] Li, Y.; Ibanez-Guzman, J.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020).

[22] Badue, C. et al.: Self-driving cars: A survey (2021)

[23] Yurtsever, E. et al.: A survey of autonomous driving: Common practices and emerging technologies (2020)

[24] Rasshofer, R. H.; Gresser, K.: Advances in Radio Science Automotive Radar and Lidar Systems (2005).

[25] Ranganathan, A. et al.: Light-weight localization for vehicles using road markings (2013).

[26] Akai, N. et al.: Robust localization using 3D NDT scan matching (2017).

[27] Krizhevsky, A. et al.: Imagenet classification with deep convolutional neural networks (2012).

[28] Deng, J. et al.: Imagenet: A large-scale hierarchical image database (2009).

Although all present methods rely on DCNNs, a differentiation can be drawn between single stage and region proposal detection frameworks. In a single stage detection framework, the classification and localization of objects is conducted by a single network. A region proposal detection framework, on the other hand, performs this task in two separate stages. First, general regions of interest are identified. Then, a classification network is used to determine the class of the objects. Many object detection challenges are currently dominated by region proposal methods. However, these algorithms require more computing power and are often more difficult to train and implement. Therefore, single-stage detectors are frequently used for automated driving tasks, as they are faster and require less memory.[29]

As the environment around us is three-dimensional, it is helpful for an ADS to have a spatial understanding of its surroundings in order to plan trajectories and navigate the vehicle safely. To obtain a 3D representation of the environment, the distance to objects must be determined. In fact, depth information can be obtained with only one camera.[30] The robustness of camera-based distance measurement can be enhanced with a stereo camera setup.[31] Unlike cameras, sensors such as radar or lidar can directly measure the distance to surrounding object. In addition, these sensors are also less susceptible to changing light or weather conditions as they actively scan the environment. However, lidar and radar sensors only collect relatively sparse 3D points, whereas cameras provide data as dense image tensors. This sparse representation of the scene poses a challenge for object detection and classification.[29] Traditional approaches group points into objects using euclidean clustering[32] or region-growing methods[33]. These clustering methods have been improved through the use of various filter techniques, like ground filtering[34] or map-based filtering[35].

Since object detection methods based on deep CNNs provide excellent results for 2D image data, it is obvious to apply them to 3D data from radar and lidar sensors. However, 3D data is not directly compatible with 2D object detection frameworks. To address this issue, various method try to replicate 2D image data in order to utilize established methods for 2D object detection and classification. Another option is to use RGB-D sensors, which combine both camera and lidar data to produce a 2D image enhanced with depth information.[36] More recent methods are able to directly process 3D data by previous applying symmetric functions to the data. PointNet, a neural network for 3D classification and segmentation, utilizes this approach to generate a feature vector which can be interpreted by a classification network.[37] 3D detection frameworks such as VoxelNet[38] or PointPillars[39] extend the basic principle of PointNet to 3D sensor data containing more than a single object. A detailed explanation of the approaches and algorithms that can be used for 3D object detection is given in chapter 3.3.

[29] Yurtsever, E. et al.: A survey of autonomous driving: Common practices and emerging technologies (2020)

[30] Ma, X. et al.: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving (2019).

[31] Cheng, X. et al.: Learning depth with convolutional spatial propagation network (2019).

[32] Rusu, R. B.: Semantic 3D object maps for everyday manipulation in human living environments (2010).

[33] Wang, W. et al.: Incremental and enhanced scanline-based segmentation method (2016).

[34] Narksri, P. et al.: A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles (2018).

[35] Lambert, J. et al.: Tsukuba challenge 2017 dynamic object tracks dataset for pedestrian behavior analysis (2018).

[36] Lai, K. et al.: A large-scale hierarchical multi-view rgb-d object dataset (2011).

[37] Qi, C. R. et al.: Pointnet: Deep learning on point sets for 3d classification and segmentation (2017).

[38] Zhou, Y.; Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection (2018).

[39] Lang, A. H. et al.: Pointpillars: Fast encoders for object detection from point clouds (2019).

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) are a type of computational model that can learn from data, recognize patterns and make decisions. These algorithms can adjust their structure and optimize their performance based on their past experiences. The chapter starts with a historical introduction to early neural networks, which is helpful for the understanding of modern neural networks by providing insights into the development of key concepts, design choices and limitations that have influenced their evolution. This context not only explains the progression to modern architectures, but also highlights the challenges that have been overcome, allowing a more informed exploration of emerging paradigms such as spiking networks, which will be discussed in more detail in chapter 2.3 and 3.4. The next subsection discusses contemporary methods of neural networking. This chapter primarily focuses on CNNs, which are a special form of neural networks capable of extracting and interpreting features from 2D or 3D images using trainable filters. This type of neural networks are a key technology for object detection and have revolutionized various computer vision tasks by enabling automated recognition, localization, and classification of objects within images and videos.

2.2.1 The Perceptron and Early Neural Networks

In 1943, Warren McCulloch and Walter Pitts laid the groundwork for artificial neurons, the building blocks of ANNs. In their seminal work "A Logical Calculus of the Ideas Immanent in Nervous Activity" they proposed a computational model of a neuron. Their threshold logic unit 2-1 fires a binary output $\sigma(x)$ when the sum of the excitatory binary inputs x_i is greater than the threshold value θ.[40] The McCulloch and Pitts neuron model is simple and flexible, but one of its main drawbacks is that it cannot learn from data. It is also unable to represent the boolean function XOR or any other non-linear function, and all inputs have the same weight.

$$\sigma(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i > \theta \\ 0 & \text{otherwise} \end{cases} \tag{2-1}$$

Inspired by the ability of the brain's neurons to change their synaptic strength over time during the learning process, better known as neuroplasticity, Frank Rosenblatt introduced the perceptron in 1957. The perceptron is one of the earliest forms of ANNs and demonstrated the ability to learn and distinguish between different classes of patterns using a weight-update rule. Basically, the perceptron described in equation 2-2 computes a weighted sum for a given input vector x and then passes it to a Heaviside function to generate its binary output. Even though this neuron is quite similar to the one proposed in 1943 by McCulloch and Pitts, it has some major differences. One of them is that the neuron takes an extra constant input b, which is known as the bias. Furthermore, the synaptic weights w are not uniform, so some inputs can influence the neuron's output more than others.[41] A major drawback of the perceptron is its limitation to linear binary classification problems due to its binary output.

[40] McCulloch, W. S.; Pitts, W.: A logical calculus of the ideas immanent in nervous activity (1943).

[41] Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. (1958).

$$\sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2-2}$$

In 1960, Widrow and Hoff improved the perceptron by using a continuous learning rule, a linear activation function, and error gradient minimization based on the squared error function. These improvements allowed the model, known as Adaline, to be used for more complex problems where linear classification is insufficient. In addition, Adaline can be used to solve regression problems by estimating continuous output rather than just binary classification.[42] However, these early artificial neural models have some limitations. In 1969, Minsky and Papert highlighted several of these limitations. Their critique centered primarily on Frank Rosenblatt's perceptron. They noted that the perceptron could only solve linear classification problem where data points could be partitioned by a basic linear decision boundary. Therefore the perceptron is unable to handle non-linear classification problems. Another point of criticism is that the perceptron consists of only one artificial neuron. Because of this limitation, the perceptron struggles to capture and adequately represent complex relations within the data. Minsky and Papert also criticized the lack of non-linear activation functions, which also limits the perceptron's ability to solve more complex problems.[43] Today, artificial neuron models therefore use non-linear activation functions. Figure 2-3 shows a diagram of a modern artificial neuron. Minsky and Papert's criticism had a far-reaching effect on research in the field of artificial neural networks and led to a sharp decline in interest in neural networks in the following years, known as the AI (Artificial Intelligence) winter.

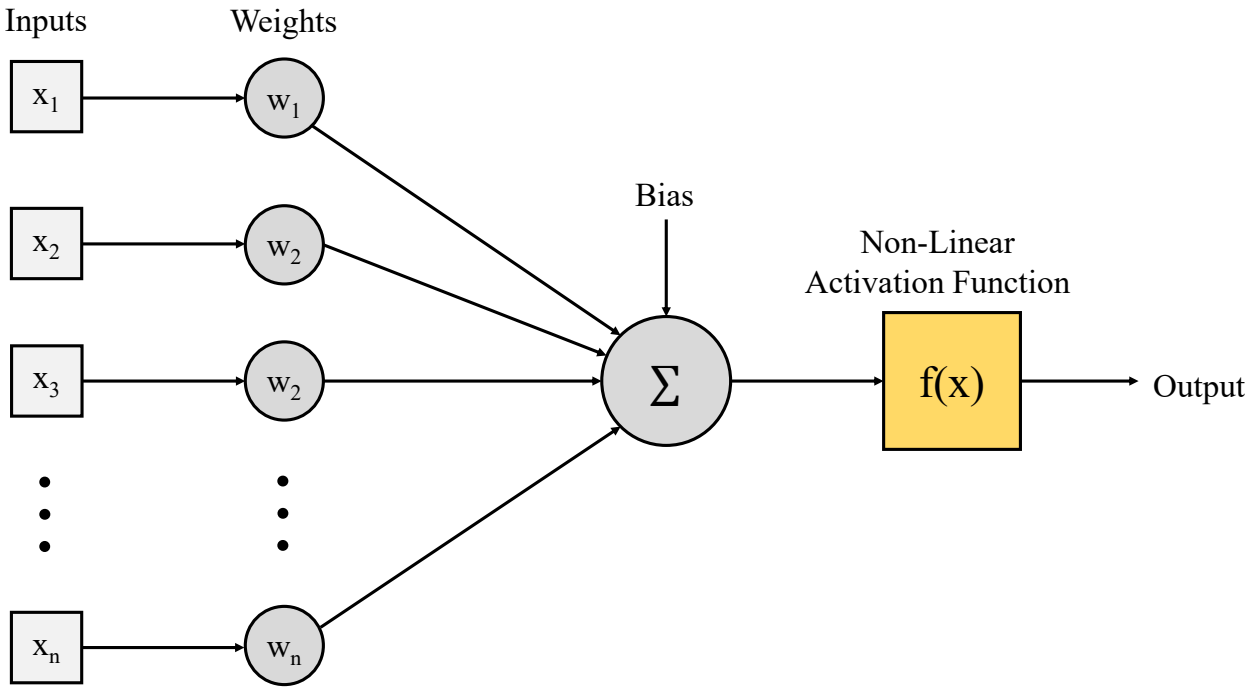

Figure 2-3: Diagram of a artificial neuron with weighted inputs, bias and non-linear activation function.

[42] Widrow, B., Hoff, M. E., et al.: Adaptive switching circuits (1960).

[43] Minsky, M.; Papert, S. A.: Perceptrons: an introduction to computational geometry (2017).

One way to represent complex relationships within the data is to increase the number of artificial neurons. ANNs with larger numbers of neurons organized into layers can learn hierarchical representations of data, allowing complex features to be extracted from raw inputs. In particular, the addition of so-called hidden layers enhances this ability. These layers are not directly connected to the input or output of the network, as shown in figure 2-4. With the learning methods discussed so far, training an ANN becomes increasingly difficult as the number of layers grows. In 1986, David Rumelhart, Geoffrey Hinton and Ronald Williams published a seminal paper on the back-propagation algorithm, which allowed ANNs with hidden layers to be trained and optimized efficiently.[44]

The back-propagation algorithm follows two steps. First, in the forward pass, the output of each layer is calculated based on the input and the activation functions. The calculated output y is compared with the actual target values $\hat{y}$ to determine the total error. The total error (TE) is defined by equation 2-3, where c is an index over input-output pairs and j is an index over the output neurons. In the second step, the backward pass, the gradients of the error function are fed back through the network using the chain rule of differential calculus. The calculated gradients are used to update the weights and bias parameters for each layer.[44]

$$TE = \frac{1}{2} \sum_{c} \sum_{j} (y_{j,c} - \hat{y}_{j,c})^2 \tag{2-3}$$

A drawback of the back-propagation algorithm is that it does not guarantee to find the global minimum of the error function. It is therefore possible to train similarly performing networks with different weights and bias parameters. In addition, this training algorithm differs from the learning behavior of biological neurons in the brain. Nevertheless back-propagation is a powerful algorithm for training large neural networks.[44]

2.2.2 Deep Learning and Convolutional Neural Networks

There are three general approaches to machine learning in the field of AI: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the model is trained on input data for which the corresponding output data is known. During the model training, the weights and bias parameters are adjusted to minimize the error between the predicted output and the actual output. Unsupervised learning involves training a model on datasets with no predetermined output labels. The goal is to discover patterns, structures or commonalities in the data without explicitly telling the model what to look for. The model attempts to segment the data into meaningful groups or clusters, or to produce a condensed representation of the data. The third type of machine learning is reinforcement learning, where an agent acts in an interactive environment. Through trial and error and experience, the agent is rewarded or punished according to his actions. The goal of the agent is to learn the best actions in order to maximize the cumulative reward. This chapter focuses mainly on models and algorithms that belong to the category of supervised learning, which is the most common form of machine learning.[45]

A significant advancement in AI is the use of deep learning.[46] The complex architecture of deep neural

[44] Rumelhart, D. E. et al.: Learning representations by back-propagating errors (1986)

[45] Mahesh, B.: Machine learning algorithms-a review (2020).

[46] LeCun, Y. et al.: Deep learning (2015).

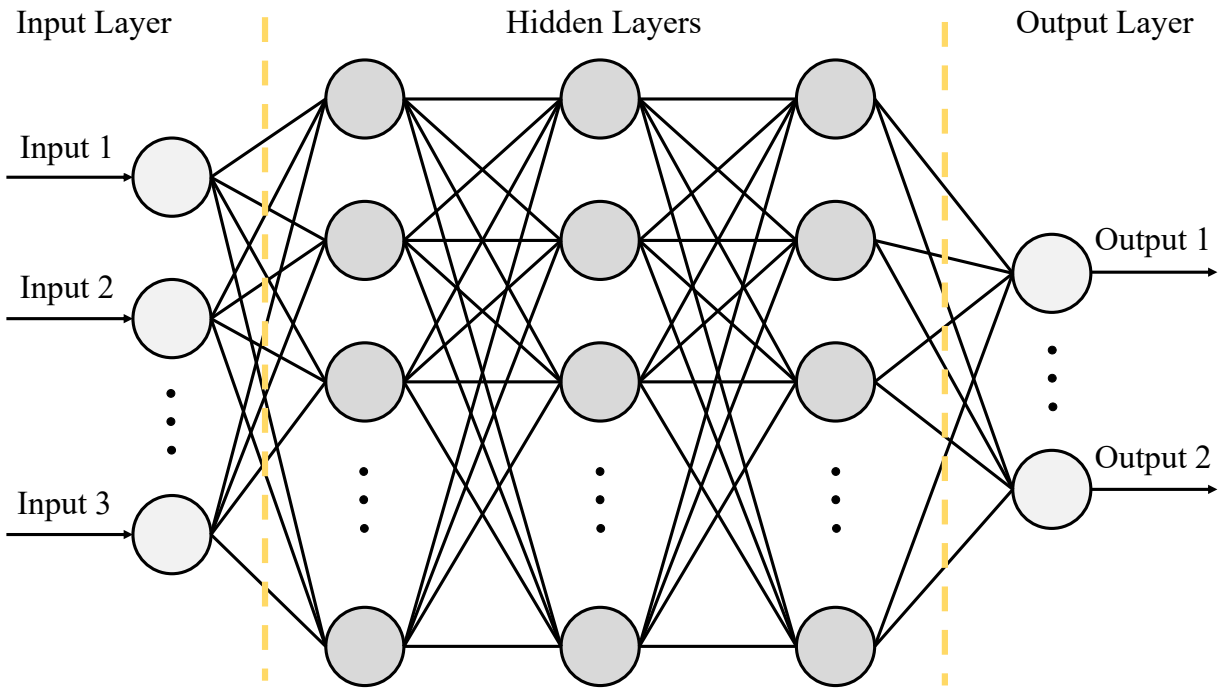

Figure 2-4: Fully connected artificial neural network with three hidden layers.

networks (DNNs), with multiple hidden layers between the input and output of the network, is one of the main reasons for their remarkable capability and performance in various domains.[47] Furthermore, DNNs have the capability to convert large datasets into a low-dimensional representation, which is useful for data compression, noise reduction or anomaly detection, as well as for automated feature extraction. This is particularly advantageous in the field of image recognition.[48] DNNs with fully connected neurons have shown exceptional efficiency in the processing of one-dimensional signals and sequences, even in the field of natural language. However, for more-dimensional data such as images or video, which are typically three-dimensional tensors, CNNs are a more sophisticated and tailored approach. These networks play a central role in modern computational perception because of their inherent ability to handle multidimensional arrays.[49]

There are several aspects that make CNNs indispensable for image recognition. One of them is the concept of shared weights. Suppose we want to use an ANN to detect features in an image. A color image is a three-dimensional tensor with the dimensions height, width and RGB channel. In order to process this type of data with an ANN, we need to flatten this tensor into a vector. There are two main problems with this approach. First, we lose the spatial information of the image by flattening the input tensor. This is undesirable because the neighboring pixels have a high feature correlation. Second, when we connect the input vector to the next layer, which has an appropriate number of neurons to detect features, we have to represent each pixel with a dedicated weight. Assuming we want to link an RGB image with a modest resolution of 32 x 32 to a layer of the same size, our ANN would have over 3 million trainable parameters.

[47] Montufar, G. F. et al.: On the number of linear regions of deep neural networks (2014).

[48] Hinton, G. E.; Salakhutdinov, R. R.: Reducing the dimensionality of data with neural networks (2006).

[49] LeCun, Y. et al.: Deep learning (2015).

However, if we combine this image with the next layer using a convolution filter of size 5 x 5 x 3, we get a CNN with only 75 weights. CNNs therefore consider local regions in the image rather than the whole image represented by individual pixels.[50] This is done by moving a filter matrix with learned weights over the whole image. This filtering operation is know as convolution which is why the corresponding layer is called convolutional layer. The major advantage of this approach is that the weights of a filter are shared as it is moved across the image, which significantly reducing the number of trainable parameters.[51] It should be noted that the weights of the filters are commonly trained via supervised learning, which is one of the key differences regarding earlier approaches that primarily used hand-written filters.[50] Furthermore convolution enables to recognize features regardless of their position in the image, because the weights of the filter are independent of the location in the image.[51]

The result of the convolution is organized in feature maps, where each feature map is assigned to a specific filter. The weighted sum is than passed through a non-linear activation function like the rectified linear unit (ReLU).[52] The ReLU activation function enhances the speed of the training process in neural networks featuring several hidden layers. In contrast, smoother non-linear activation functions, such as tanh or sigmoid, can demand an unsupervised pre-training of the CNN.[53] Another argument for ReLU is that activation functions with a constant derivative prevent the network from becoming trapped in a saddle point or a poor minimum during training.[54]

Another key aspect in the CNN architecture is the so-called pooling, which follows the activation function. The pooling layer is used to reduce the dimension of the feature map while keeping the important information. Pooling also ensures that the network is less sensitive to minor shifts in the input image. This increases the network's ability to detect patterns regardless of their exact position. The most common form of pooling is max-pooling, where the highest value in a given range is retained. Generally, multiple stages consisting of convolution, non-linear activation and pooling are stacked. This structure allows the CNN to learn high level features from the combination of low level features. In the first layers of the network, simple features such as edges are recognized and then combined into more complex features and motifs in the following layers. These high level features form the output of the feature extractor, which is also often referred to as backbone. Based on the output of the backbone, several computer vision tasks can be performed, such as image recognition, object detection or semantic segmentation.[52] Figure 2-5 illustrates an exemplary structure of a CNN with its different layers. In this figure, the output of the backbone is first flattened in order to process the extracted features with an ANN.

A so-called soft-max layer is usually used at the end of the neural network. This layer calculates a confidence score for the detected object classes in a given image. As input, the soft-max layer receives the activation values of the previous layer x_i as a vector. An exponential function is first applied to these values, whereby positive values are emphasized and negative activation values are suppressed. To obtain the probability distribution P, this result is normalized by dividing it by the sum of all exponential values. The prediction is than assigned to the class with the largest probability.[55]

[50] Albawi, S. et al.: Understanding of a convolutional neural network (2017)

[51] Zhao, Z.-Q. et al.: Object detection with deep learning: A review (2019)

[52] LeCun, Y. et al.: Deep learning (2015)

[53] Glorot, X. et al.: Deep sparse rectifier neural networks (2011).

[54] Fahlman, S. E. et al.: An empirical study of learning speed in back-propagation networks (1988).

[55] Wu, Y. et al.: Deep convolutional neural network with independent softmax for large scale face recognition (2016).

$$P(y_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2-4}$$

Depending on the task, we want to classify not only one but several objects and determine their position in the image. This is referred to object detection, which aims to label objects and mark there position with rectangular bounding boxes. As already mentioned in chapter 2.1.2, there are mainly two approaches to this task. One is inspired by traditional object detection pipelines and first generates region proposals and classifies each proposal. The second approach treats object recognition as a global regression and classification problem. These single stage frameworks compute bounding boxes and class confidence in parallel.[56] At this point we want to take a closer look at two concepts that are common to both approaches.

Both frameworks return proposed bounding boxes and class confidence scores for each bounding box. One problem that arises is that objects are usually detected multiple times. Therefore, they are not marked with a unique bounding box. To address this problem, a post-processing algorithm known as non-maximum suppression (NMS) is responsible for selecting the best-fitting bounding box of an object. Standard NMS algorithms first sort the bounding boxes based on their confidence score and then remove any bounding boxes that have a significant overlap, also referred to as intersection over union (IoU), with the highest scoring proposal. This process is repeated until there is no significant overlap.[57] Most detection frameworks use a greedy NMS. The difference is that greedy NMS only performs a single overlap check for each selected bounding box and does not perform any further iterations to determine the exact number of bounding boxes to keep. This results in faster calculations of unique bounding boxes, but there is also a

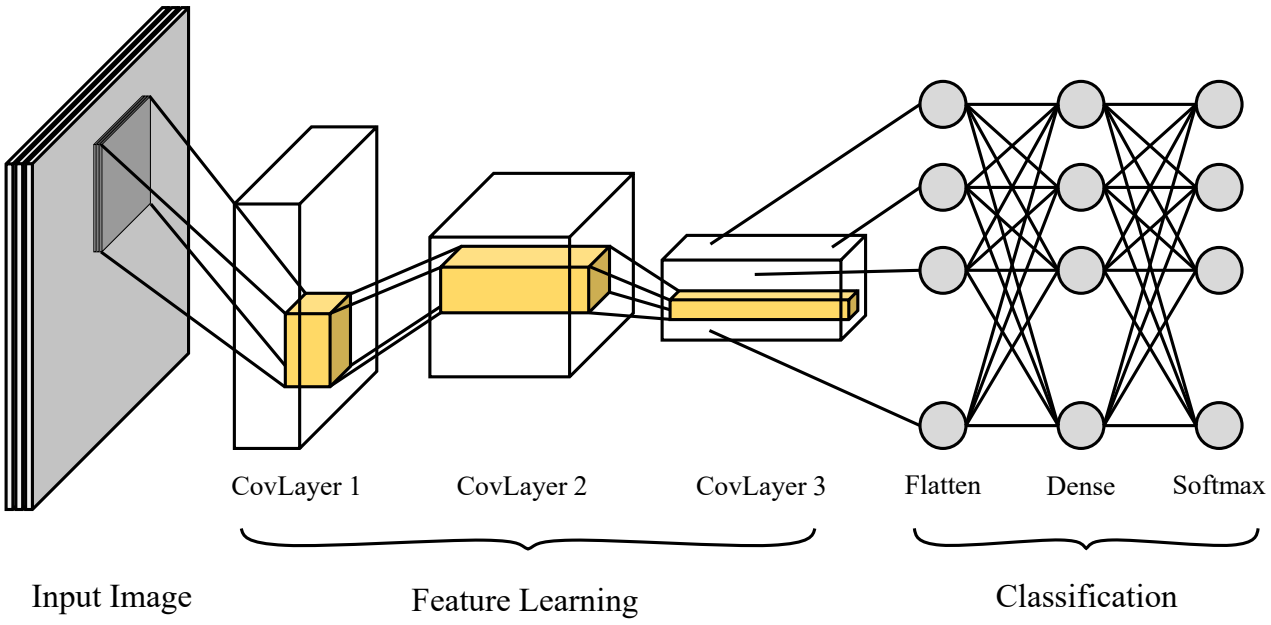

Figure 2-5: Architecture of a CNN for image classification with three convolution layers, flatten layer and soft-max layer.

[56] Zhao, Z.-Q. et al.: Object detection with deep learning: A review (2019).

[57] Neubeck, A.; Van Gool, L.: Efficient non-maximum suppression (2006).

risk that some valid bounding boxes may be discarded, especially if multiple objects overlap heavily. By default, standard and greedy NMS decide whether or not to remove a proposed bounding box based on a fixed parameter that controls how wide the suppression is. To overcome this hard-coded nature of NMS, efforts are being made to integrate NMS into the end-to-end learning process of the framework.[58]

Another concept shared by region proposal detectors and single stage detectors are anchor boxes. The main task of anchor boxes is to facilitate the prediction of bounding boxes for different objects in an image. For this purpose, predefined bounding boxes with different sizes and shapes are distributed across the image and act as suggestions for regions of interest. This reduces the computational effort considerably, as only the proposed anchor boxes are analyzed in more detail.[58] The YOLOv2 [59] single stage detector and the Faster R-CNN [60] region proposal detector were among the first to use anchor boxes. More recent approaches, such as YOLOv5 [61], use an algorithm to adjust the predefined anchor boxes depending on the dataset used. The model is then trained with the new anchor boxes, which are more appropriate for the given dataset.

2.3 Spiking Neural Networks

Spiking neural networks (SNNs) are a type of neural network model inspired by the biological functioning of the brain. Unlike traditional ANNs, which use floating point operations (FLOPS), SNNs operate with discrete-valued and time-dependent spikes or pulses of activity. These spikes are analogous to the firing of neurons in the human brain. SNNs use the temporal information of these spikes to process and transmit information, allowing them to capture temporal dynamics and potentially achieve greater efficiency and biological plausibility compared to ANNs. However, training and implementing SNNs presents unique challenges due to the discrete nature of spikes and requires specific learning algorithms and hardware implementations.[62] To understand the similarities between SNNs and the human brain, we first discuss the structure of biological neurons and how they transmit information. We then review the spike response model (SRM), which is able to mimic the phenomenological functioning of various biological neurons. Further models of spiking neurons are introduced in chapter 3.4. The final section presents two fundamental approaches for the encoding of floating point numbers into discrete spikes. This pre-processing step is usually necessary because most data, including sensor data, are floating point numbers and therefore cannot be processed directly by SNNs. Further encoding schemes are discussed later in chapter 3.2.

2.3.1 Biological Neurons

The human brain contains around 80 billion neurons that intricately link in a vast network of trillions of connections. About three quarters of these neurons are found in the heavily creased cerebral cortex. One third of the cortex is involved in visual processing. Additionally, the brain is partitioned into two hemispheres along its longitudinal axis, with the dominant hemisphere controlling abilities like reasoning, speaking, and writing. The non-dominant hemisphere is responsible for emotional and holistic thinking.

[58] Hosang, J. et al.: Learning non-maximum suppression (2017)

[59] Redmon, J.; Farhadi, A.: YOLO9000: better, faster, stronger (2017).

[60] Ren, S. et al.: Faster r-cnn: Towards real-time object detection with region proposal networks (2015).

[61] Jocher, G. et al.: ultralytics/yolov5: v3. 0 (2020).

[62] Tavanaei, A. et al.: Deep learning in spiking neural networks (2019).

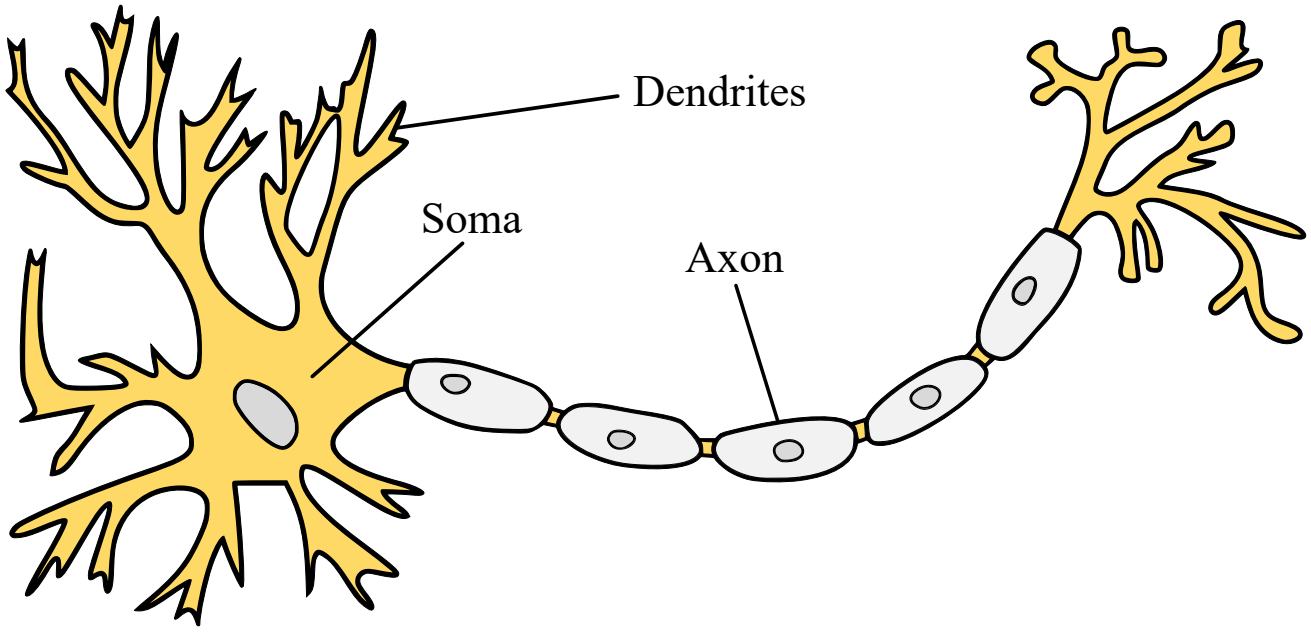

Figure 2-6: Illustration of a biological neuron consisting of dendrites, soma and axon.

While certain tasks can be assigned to particular parts of the brain, these areas are strongly interconnected and cannot be considered separately.[63] All areas of the brain comprise neurons, which are the fundamental units for processing information. In addition to different types of nerve cells, there are also glial cells that support the neurons. Although they do not play a direct role in information processing, they are necessary for energy supply and make up a significant proportion of brain tissue.[64] A single neuron, as shown in figure 2-6, consists of three basic parts called dendrites, soma, and axon.

The dendrites receive signals transmitted from connected neurons and conduct them to the soma. The soma performs a non-linear processing and outputs a signal along the axon when the input exceeds a certain threshold. The sending neuron is referred to as the presynaptic cell, while the receiving neuron is called the postsynaptic cell. The junction between two neurons is called a synapse. A single neuron is often connected to several thousand postsynaptic cells, which are usually in close proximity. Neurons transmit information through action potentials, also known as spikes. These spikes are binary events and their shape has no influence on the information transmitted. Rather, the number and precise timing of the spikes contain all the information. Often, a series of spikes occur at short intervals, known as a spike train. Although spikes can occur in rapid succession, once a spike has been sent, neurons enter a phase during which they are unable to send another spike, even under high stimulation. This is referred to as the absolute refractory period, which represents the minimum time interval required for subsequent spikes to occur. This phase is then followed by a period of relative refractoriness, during which a stronger stimulus is required to excite the neuron once again. When an action potential reaches the synapse, neurotransmitters from the presynaptic cell are released. Specialized receptors in the postsynaptic cell detect these neurotransmitters and allow an ion influx. The ion flux, in turn, leads to a change in the membrane potential of the postsynaptic cell. This response is also referred to as the postsynaptic potential.[64]

[63] Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019), pp. 87-90.

[64] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 11-15.

This information processing heavily depends on the strength and adaptability of synaptic connections. These connections can be modified through learning and experience. This property is called synaptic plasticity, which allows the brain to adapt to new information and build memories. Synaptic plasticity is a fundamental mechanism that enables the brain to achieve its remarkable potential for cognition and learning.[65] All this remarkable processing power consumes about 20% of the total energy used by the human body.[66] Based on a basal metabolic rate of 1700 kcal per day for an average human, the brain consumes less than 17 watts of power. Since the energy demand of neurons is proportional to their surface area, it is estimated that the energy demand of the neocortex, a part of the cerebral cortex responsible for higher cognitive functions that has evolved in mammals[67], accounts for approximately 40%.[68] This low energy demand contrasts with the immense energy demand of conventional ANNs. If the way biological neurons process information could also be implemented in AI applications, a drastic reduction in energy demand can be assumed.

2.3.2 Spike Response Model

Now that we have a basic understanding of how biological neurons transmit information, we need a mathematical description that allows us to simulate these neural dynamics. For this purpose, we discuss the spike response model (SRM), which provides a phenomenological description of biological neurons. However, we first want to take a closer look at the electrical potentials that lead to the occurrence postsynaptic action potentials in order to better comprehend the mathematical equation of the SRM.

The difference in electrical charge between the inside of a neuron and its surroundings is called the membrane potential. The membrane potential of an unexcited neuron is know as the resting potential and has a negative charge of approximately -65 mV. Changes in membrane potential are caused by postsynaptic potentials (PSPs), which can have an excitatory (EPSP) or inhibitory (IPSP) effect on the membrane potential. These PSPs result from the release of neurotransmitters at the synapse, triggered by action potentials from presynaptic neurons. When the membrane potential of a neuron reaches a certain threshold, which is typically around -45 mV to -35 mV, an action potential with an amplitude of approximately 100 mV is emitted along the axon of the postsynaptic neuron. This action potential, in turn, can lead to the release of neurotransmitters at neighboring synapses, resulting in PSPs that excite neighboring neurons. In this way, information is spread across many neurons in the brain.[69]

The number of PSPs and their timing are decisive for whether the membrane potential of the neuron reaches the threshold and an action potential is triggered. A single PSP has an amplitude of about 1 mV, whereas at least 20 PSPs are required to reach the threshold of the neuron to fire an action potential. As the membrane potential decreases with time, the postsynaptic potentials must arrive at the neuron in rapid succession. If the time interval between stimuli is greater, more PSPs are required to cross the threshold. If too few or no PSPs reach the neuron, the threshold is not reached or the membrane potential returns to the resting potential. After an action potential has been sent, the neuron enters a period of refractoriness

[65] Citri, A.; Malenka, R. C.: Synaptic plasticity: multiple forms, functions, and mechanisms (2008).

[66] Mink, J. W. et al.: Ratio of central nervous system to body metabolism in vertebrates (1981).

[67] Creutzfeldt, O. D.: Generality of the functional structure of the neocortex (1977).

[68] Lennie, P.: The cost of cortical computation (2003).

[69] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 14-17.

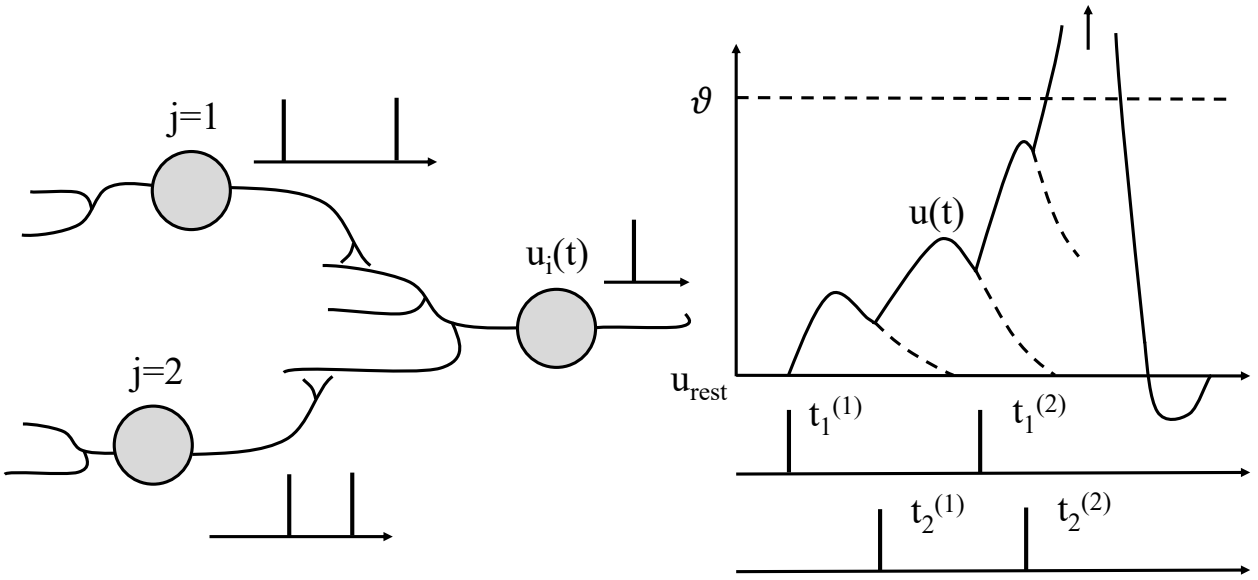

Figure 2-7: Change of the membrane potential of the SRM due to incoming spikes of two neighboring neurons.[70]

during which a higher stimulus is required to re-excite the neuron.[70] Figure 2-7 shows the change in membrane potential of a neuron receiving input signals from two other neurons.

One model for simulating neuronal dynamics is the spike response model (SRM). The simplest form of the SRM expresses the membrane potential $u_i(t)$ of a neuron using equation 2-5. For simplicity, the resting potential u_{rest} is usually set to zero, although the actual resting potential of biological neurons may vary depending on neuron type and state. The ϵ term describes the change in membrane potential in response to incoming spikes at the soma of the postsynaptic cell and is referred to as the synaptic kernel. The reaction of the firing neuron to its own spikes is defined by the refractory term η. This kernel describes the absolute and relative refractory periods. When the membrane potential reaches the threshold ϑ as in equation 2-6, a spike is fired by the SRM neuron.[71]

$$u_i(t) = \eta(t - t_i) + \sum_j \sum_f \epsilon_{ij}(t - t_j^{(f)}) + u_{\text{rest}} \tag{2-5}$$

$$u_i(t) = \vartheta \tag{2-6}$$

It is important to note that the shape of the action potential usually does not contain essential information, so the actual shape of a spike is often approximated by a Dirac function to simplify calculations.[72] We are mainly interested in the phenomenological description of biological neurons. Therefore, the SRM

[70] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 14-17.

[71] Gerstner, W.: Time structure of the activity in neural network models (1995).

[72] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 19.

describes the timing of the spike generation, rather than the fundamental mechanisms that lead to the generation of action potentials.

Although this model attempts to reproduce the complex dynamic processes of a biological neuron, it is subject to certain simplifications. For example, the SRM takes only the most recent spike into account. If we stimulate a biological neuron with a constant current, it will respond with a series of spikes. However, the intervals between the spikes increase over time until a constant firing period is reached. This response is known as adaptation and most biological neurons show this behavior, which is why they are called regularly firing neurons. Since adaptation is a temporal process and the SRM only considers the most recent spike, it is unable to mimic the adaptation of regular spiking neurons. Another class of biological neurons are bursting neurons, which response to a constant stimulus with a series of spikes interrupted by intervals of no activity.[73] Other biological neurons respond with a post-inhibitory rebound when a negative current is turned off and emit one or multiple rebound spikes. Bursting and inhibitory rebound spikes of biological neurons cannot be simulated by the SRM either.[74]

This brief introduction highlighted the challenges in modeling neuronal dynamics, given the vast variety and the time dependent dynamics of biological neurons. In chapter 3.4 we discuss spiking neuron models that are simplifications of the SRM, such as the integrate-and-fire model, but also models that attempt to model the dynamic behavior of biological neurons more accurately, such as the Hodgkin-Huxley model.

2.3.3 Data Encoding

As SNNs use discrete spikes instead of floating point numbers to represent information, the transformation of the input data is typically a fundamental step of a SNN implementation. There are various encoding schemes to represent floating point data as spike patterns. The main categories of encoding include rate and temporal coding, which both result in different spike properties.[75] Traditionally, rate coding has been the most common principle, in which information is encoded via the average firing rate of the neuron. The most basic rate encoding scheme is rate count, where the frequency of the spikes corresponds to the magnitude of the input value. Usually, higher values in the input data result in higher firing rates, and lower values result in lower firing rates of the neuron model. Rate coding is simple to implement and can represent floating point numbers as a series of spikes by using different firing rates. The mean firing rate is calculated from the number of spikes divided by a predefined time window.[76] In 1926, the concept of average firing rate was first explored using stretch receptor neurons. In this experiment, muscle fibers of a frog were subjected to a force, resulting in an observed change in the average firing rate of the neurons.[77]

A major disadvantage of rate coding is the loss of temporal information due to averaging. Furthermore, rate coding is only suitable for applications in which the stimulation of the neurons occurs at a constant and slowly changing level. This limits this type of encoding as most real world data is not stationary.[78]

[73] Connors, B. W.; Gutnick, M. J.: Intrinsic firing patterns of diverse neocortical neurons (1990).

[74] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 19-20.

[75] Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019), p. 128.

[76] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 24-25.

[77] Adrian, E. D.; Zotterman, Y.: The impulses produced by sensory nerve-endings (1926).

[78] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 26.

Also, the human ability to rapidly perceive the environment in under 100 ms cannot be achieved through rate coding, as the typical averaging time window is already a few hundred milliseconds.[79]

Another popular approach, where the precise timing of spikes is taken into account, is temporal coding. The most basic scheme is time-to-first spike (TTFS) coding, in which the information is determined by the time difference between the onset of stimulation and the first spike of the neuron.[80] In its purest version of this scheme, each neuron only emits a single spike. The neuron can only transmit information again at the onset of the next stimulation. Unlike rate coding, this type of encoding represents information solely by the timing of the spikes, not by the number of pulses. In this way, abrupt changes in the input can be well represented. These changes in input occur, for example, in visual observation of the environment as the gaze moves from one object to the next.[81] Although TTFS coding represents an idealised representation of information, it has been shown that the first 20 ms to 50 ms of a neural signal contain a high level of information.[82]

The nature of the given task and the significance of the temporal dynamics in the data to be processed determine the choice between rate and temporal encoding. For the sake of spike sparsity, it is preferable to represent information with as few spikes as possible. However, a efficient encoding of the incoming data does not necessarily result in an efficient system. Furthermore, the architecture of the perception pipeline must be aligned with the choice of the encoding scheme.[80] Chapter 3.2 therefore discusses various subcategories of encoding schemes that are based on rate and temporal coding. In this chapter, the potential advantages and disadvantages of different coding techniques are examined in detail, with a focus on accuracy, dynamic behavior and latency, among other factors.

[79] Keysers, C. et al.: The speed of sight (2001).

[80] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021)

[81] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 30.

[82] Tovee, M. J. et al.: Information encoding and the responses of single neurons in the primate temporal visual cortex (1993).

3 State of Literature and Research

This chapter presents a comprehensive exploration of the various aspects and components required to realize a neuromorphic perception pipeline, with a focus on current state-of-the-art research. The first section discusses different lidar designs, some of which are used in the automotive industry for automated driving. The basic approaches to laser-based distance measurement are described and the advantages and disadvantages of each approach are explained. In the following section, encoding of floating point data is explored in detail. Within the main categories of rate and temporal encoding, additional coding techniques are presented. The third section deals with the topic of 3D object detection based on point clouds. First, several benchmark datasets containing lidar data for automated driving are presented. Then, three basic approaches for 3D object detection based on point clouds are analyzed. The next section introduces further spiking neuron models, which can be used as basic building blocks for SNNs. In addition to the description of the individual models, they are evaluated with regard to their biological plausibility and computational complexity. In the following section, training algorithms for SNNs are presented. Due to the discrete nature of spiking neurons, the learning principles for ANNs cannot be readily applied to SNNs. This section therefore discusses various approaches that can be used to overcome this problem. The sixth section deals with the technological challenges of hardware implementation of SNNs and introduces different types of hardware specifically designed for the operation of ANNs and SNNs. The last section presents existing concepts and approaches for object detection with SNNs based on lidar data.

3.1 Lidar Sensors

Lidar sensors, short for light detection and ranging, are sensors that use laser beams to measure the distance to objects in order to create detailed 3D maps of their surroundings. Lidars are active sensors, which means they emit energy in the form of light and capture the reflected energy to gain information about the environment. In contrast, passive sensors rely on the detection of energy emitted by natural sources, such as visible light, which can be collected by cameras.[83] Lidars typically use one or multiple laser beams to scan an area of the environment known as the field of view (FoV). A photodetector then converts the reflected laser beams into points that represent both the distance and the reflectivity of the reflecting surface. These points collectively form what is known as a point cloud, with each point being associated with specific x, y, and z coordinates.[84]

Figure 3-1 shows the schematic structure of a lidar sensor, consisting of a laser modulation unit, an optional beam steering system, a photodetector, optics and a signal processing unit. Lidars can be further subdivided into the scanning system and the laser rangefinder system. The laser rangefinder consists of the laser transmitter, photodetector and signal processing electronics and emits the laser beams and processes the reflected beams. The scanning system is responsible for steering the laser beams according to a specific beam pattern with different azimuths and vertical angles, thus determining the FoV of the lidar sensor, and consists of the beam steering system and the optics.[84]

[83] Behroozpour, B. et al.: Lidar system architectures and circuits (2017).

[84] Li, Y.; Ibanez-Guzman, J.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020)

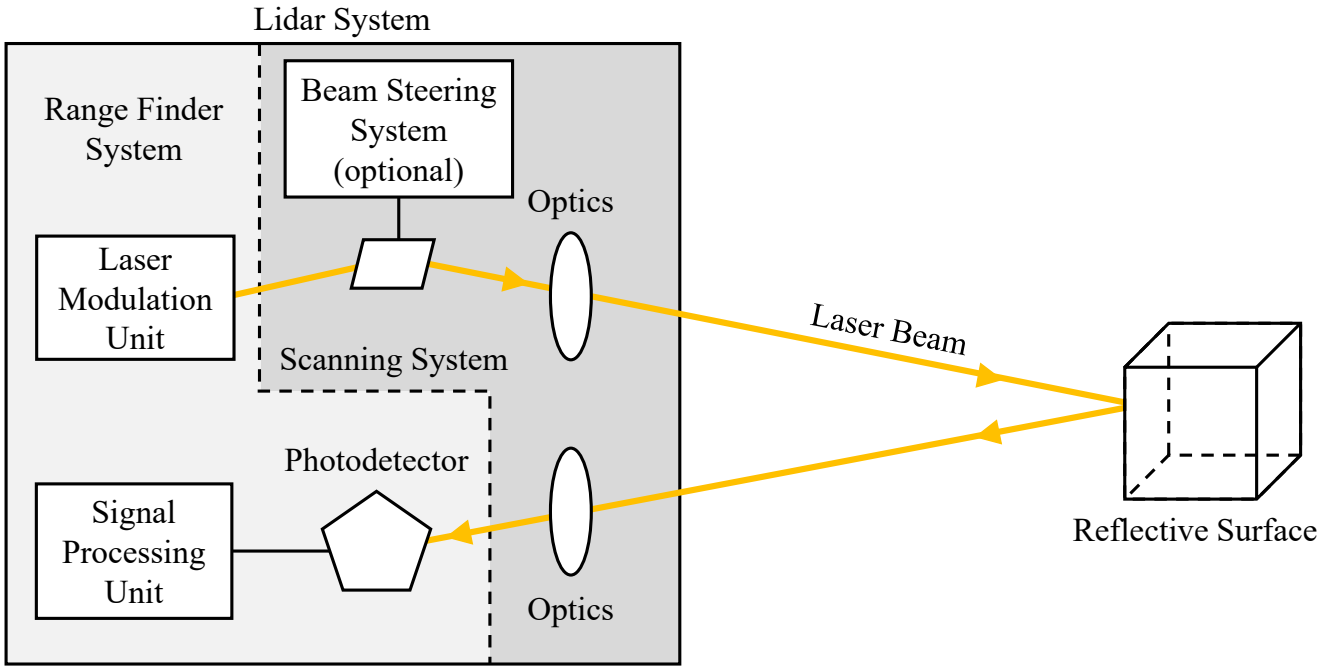

Figure 3-1: Diagram of a lidar sensor subdivided into the range finder system and the scanning system.[85]

The laser rangefinder is the essential component for measuring the distance to the objects, which can be done using pulsed lasers or FMCW (Frequency Modulated Continuous Wave) lasers. Depending on these two approaches, the rangefinder system is referred to as a direct or coherent detection laser rangefinder. Direct detection laser rangefinders determine the distance d by measuring the time Δt it takes for a pulsed laser beam to travel to an object and return which is know as the time of flight (ToF). Since the speed of light c is known the distance can be calculate according to equation 3-1.[86]

$$d = \frac{c \cdot \Delta t}{2} \tag{3-1}$$

ToF lidars are widely used in the automotive industry due to their simplicity and straightforward signal processing. However, the maximum detection range of these devices is limited by eye-safety standards, as they need to emit high-energy pulses to receive enough photons from distant objects. In addition, pulsed lidars are susceptible to interference from strong sunlight and neighboring ToF lidars. FMCW lidars, on the other hand, rely on the Doppler effect to measure both distance and velocity by using a continuously frequency modulated laser signal. This approach allows indirect measurement of distance and velocity of the objects. Theoretically, FMCW lidars can have a greater detection range than ToF lidars because they emit a continuous signal rather than high-energy laser pulses, which means they are not as limited by eye-safety standards. Also FMCW lidars are less susceptible to interference and are therefore preferred in scenarios where reliability is of crucial importance. When the frequency of the light is modulated linearly, the delay between the collected light and the source causes a constant frequency difference f_d. This

[85] Behroozpour, B. et al.: Lidar system architectures and circuits (2017)

[86] Li, Y.; Ibanez-Guzman, J.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020).

frequency difference between the emitted and received light results in a periodic phase difference which causes an alternating constructive and deconstructive interference pattern. Accordingly, the distance to an object can be determined using equation 3-2, with γ representing the slope of the frequency modulation.[87]

$$d = \frac{c \cdot \Delta f}{2\gamma} \quad \text{with} \quad \gamma = \frac{\Delta f_{\max}}{\Delta t} \tag{3-2}$$

Most lidar sensors use lasers that emit electromagnetic radiation in the non-visible range. Depending on the application either near-infrared (NIR) or short-wave infrared (SWIR) lasers can be used. The power of the laser depending on its frequency is regulated by the eye-safety standard IEC 60825.[88] While SWIR lasers allow greater detection range due to higher allowable power levels, they use expensive indium gallium arsenide-based photodiodes, which are less efficient. Additionally, the emitted light of SWIR lasers with a wavelength of 1550 nm is more susceptible to electromagnetic absorption caused by atmospheric water vapor. Therefore, NIR lasers that use a wavelength between 850 nm and 950 nm are more common for automotive lidar systems. The reflected laser beams are converted into an electrical voltage by a photodetector, allowing further processing of the light signal. The choice of the photodetector is closely related to the wavelength of the laser.[89]

There are various scanning techniques to steer laser beams of the lidar sensor within their FoV. The different scanning approaches can be broadly classified into mechanical spinning systems and solid-state systems. Mechanical spinning systems rely on rotating mirrors or prisms to steer the laser beams. Although effective, such systems require a significant amount of installation space and are prone to vibrations, which can cause issues in automotive applications. More sophisticated lidar systems attempt to minimize the effect of vibration on the sensor by using multiple laser beams to limit the movement required of the scanning system.[89]

To further limit the movement of the scanning system, laser beam steering can be facilitated by micro-electro-mechanical systems (MEMS). MEMS technology allows the laser beam to be steered by microscopic mirrors mounted on a silicon chip. To achieve this, the mirrors are balanced between an electromagnetic force generated by a conductive coil around the mirror and an elastic force from a torsion bar, which also acts as a rotation axis. Although MEMS lidars still contain moving parts they can be seen as a near-solid-state beam steering system. This technology offers unique capabilities, such as the ability to dynamically adjust the FoV to focus the laser beams on critical areas. In addition, costs can be reduced by using silicon manufacturing techniques to produce the MEMS mirrors.[90]

Another design approach are flash lidars, which utilizes an optical diffuser lens to disperse a single laser beam across the entire FoV. Since this system scans the environment using a single laser pulse, the image frame rate is significantly higher than lidars that rely on mechanical scanning systems. The returning laser beams are captured by a two-dimensional photodetector. Originally developed for spacecraft aviation, flash lidars are truly solid state, which make them insusceptible to vibrations. Nonetheless, flash lidars

[87] Behroozpour, B. et al.: Lidar system architectures and circuits (2017).

[88] Commission, I. E. et al.: Safety of laser products-Part 1: Equipment classification and requirements (2007).

[89] Li, Y.; Ibanez-Guzman, J.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020)

[90] Yoo, H. W. et al.: MEMS-based lidar for autonomous driving (2018).

typically have a restricted detection range as a single laser beam must illuminate the whole scene at once. Furthermore, the optical diffuser limits the FoV of the sensor.[91]

Another true solid-sate design are optical phased-array (OPA) lidars. Similar to phased array radar systems, OPA lidars achieve beam steering through precise phase modulation of laser beams using an array of phase modulators. By adjusting the phase of each laser beam individually, OPA lidars can control the direction and focus of the emitted light. The manipulated laser beams combine to form an optical wavefront. This wavefront represents the spatial distribution of the laser beams and can be used to scan the environment within the FoV. Although OPA technology is considered promising for lidar sensors, there are currently no OPA lidars available on the market.[92]

In general lidars provide highly reliable range measurement which is particularly helpful for automated driving. In the automotive industry there is a clear trend towards cost reduction and improved robustness. FMCW lidars are gaining traction compared to the widely used ToF lidars as car manufacturers seek to improve their performance and reliability in various applications, moving towards a safer and more automated driving experience.[93] However, lidars have certain disadvantages compared to other sensors. Compared to cameras, lidars are more expensive and require more installation space.[94] Although lidars are superior to cameras in low-light conditions, their detection performance can suffer in severe weather conditions such as heavy rain or fog.[95] This is why the most ADSs rely on a multi-sensor perception to balance the advantages and disadvantages of different sensors and to provide reliable data in all traffic situations and weather conditions.[96]

However, lidar technology has been seen as an important component in achieving the goal of fully automated driving since the top three teams in the 2007 DARPA Urban Challenge relied on lidar as a key sensor in their vehicles.[97,98,99] This competition, in which automated vehicles had to complete a 60 mile long urban course in less than 6 hours, demonstrated for the first time that automated driving is truly possible and that lidar is a key sensor for perceiving information about the environment. Despite their high price, lidar sensors are now used in many highly automated vehicles. This is due to their high accuracy in measuring distance. However, automated systems based solely on lidar sensing are not common, as classifying objects with lidar using traditional methods is a challenging task, so many ADS use cameras in addition to provide a reliable source of semantic information. Another option is to use deep learning approaches to extract semantic information directly from the lidar data.[93] The steps required to apply deep learning to lidar sensor data are discussed in more detail in chapter 3.3.

[91] Amzajerdian, F. et al.: Imaging flash lidar for autonomous safe landing and spacecraft proximity operation (2016).

[92] McManamon, P. F. et al.: Optical phased array technology (1996).

[93] Li, Y.; Ibanez-Guzman, J.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020)

[94] Paula Veronese, L. de et al.: Evaluating the limits of a LiDAR for an autonomous driving localization (2020).

[95] Wallace, A. M. et al.: Full waveform LiDAR for adverse weather conditions (2020).

[96] Wang, Y. et al.: Multi-modal 3d object detection in autonomous driving: a survey (2023).

[97] Urmson, C. et al.: Tartan racing: A multi-modal approach to the darpa urban challenge (2007).

[98] Montemerlo, M. et al.: Junior: The stanford entry in the urban challenge (2008).

[99] Bacha, A. et al.: Odin: Team victortango's entry in the darpa urban challenge (2008).

3.2 Encoding Schemes

Building on the chapter 2.3.3, this section examines more specific encoding techniques that can be used to convert floating point numbers into spike patterns. Generally all encoding schemes can be assigned either to rate or temporal encoding. Even population codes, which are often referred to as third category of encoding techniques can be used in rate and temporal coding.[100] Starting with rate codes which can be subdivided into count, density and population rate encoding as depicted in figure 3-2. Theoretically, it is possible to determine the spike rate with only two spikes by using so-called regular spike trains, in which the time interval between the individual spikes corresponds to the reciprocal of the spike rate. However, this type of rate coding would be susceptible to noise, since already one additional spike or one missing spike would change the rate significantly. To increase the robustness of rate coding, the average spike rate is usually determined over a longer period of time.

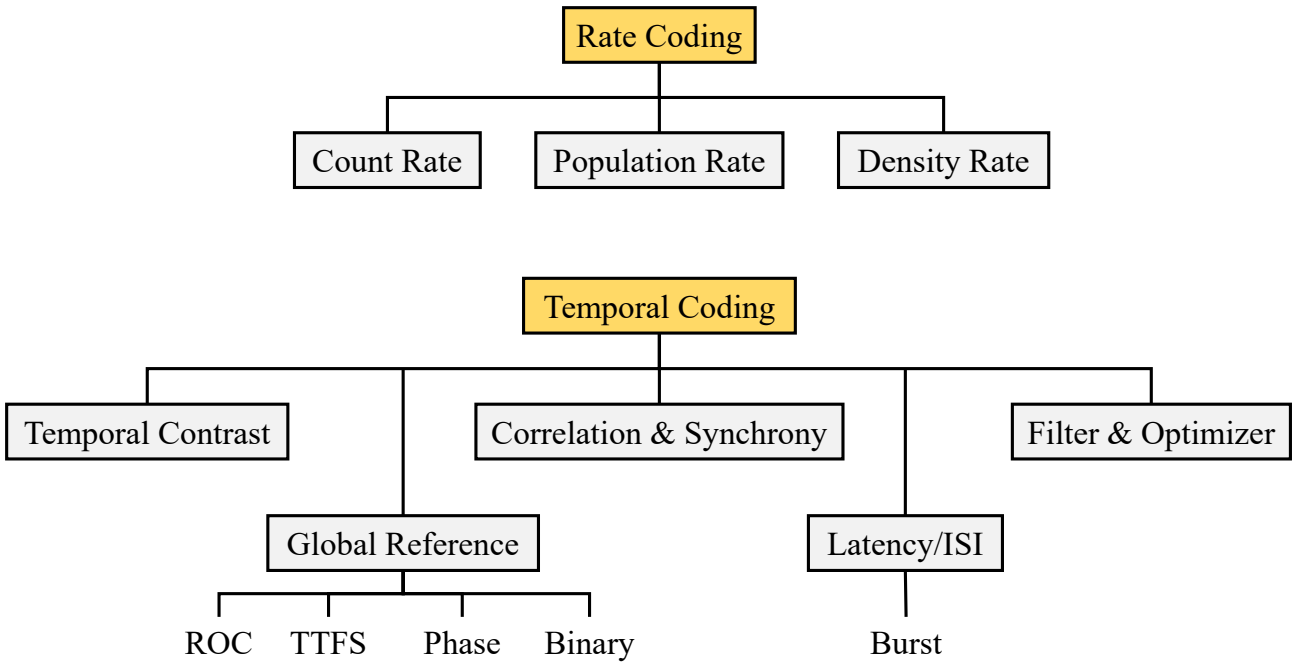

Figure 3-2: Categorization of temporal and rate coding schemes.[100]

We briefly mentioned count rate coding in the previous chapter on data encoding, which is one of the most common rate coding schemes. To obtain the count rate, we determine the average number of spikes over time, which is defined by the mean firing rate v in equation 3-3. Therefore we need the number of spikes N_{spike} within a defined time window Δt. With this kind of encoding we can only approximate floating point values. The discretisation error decreases as the number of spikes increases. The count rate is particularly suitable for the representation of constant or slowly varying values.[101]

$$v = \frac{N_{\text{spike}}}{\Delta t} \tag{3-3}$$

[100] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021)

[101] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 25-26.

Another approach defines the spike rate as a spike density. Considering the response of a neuron that is stimulated several times with the same input sequence, the spike density can be determined by the average number of spikes over all runs. The neuronal response of the runs is recorded with a peri-stimulus time histogram (PSTH), where the spike density $p(t)$ is calculated with equation 3-4. First, the number of spikes occurring in all repetitions $N_{\text{spike}}(t; t + \Delta t)$ is summed up and divided by the number of runs N_{run}. Subsequently dividing by the time interval Δt leads to the spike density for the PSTH. The result of the PSTH can be smoothed to obtain a continuous spike rate. The spike density is a useful tool for the experimental determination of neuronal activity. However, it is not a biologically plausible coding scheme, as real stimuli do not recur, and thus no spike density can be determined.[102]

$$p(t) = \frac{1}{\Delta t} \frac{N_{\text{spike}}(t; t + \Delta t)}{N_{\text{run}}} \tag{3-4}$$

If we consider a large population of independent neurons receiving the same stimulus, rather than one neuron processing the same sequence several times, we can define the spike rate as a population activity. We can describe the population activity $A(t)$ as in equation 3-5, where N_{neuron} is the number of the observed neurons and $N_{\text{spike}}(t; t + \Delta t)$ the number spikes that occur over all neurons in the population. The population rate is insensitive to the spike response of individual neurons because we are looking at the response of an entire population of neurons. This robustness increases with the number of neurons considered. In addition, looking at multiple neurons excited by the same signal makes it easier to see patterns, whereas the response of a single neuron can be difficult to interpret. This in turn means that the properties of a single neuron cannot be inferred from patterns in the population response.[103] A visualization of the presented rate coding techniques is depicted in figure 3-3. In this diagram the neurons are stimulated by a wide signal, whose beginning and end are marked by dotted lines.

$$A(t) = \frac{1}{\Delta t} \frac{N_{\text{spike}}(t; t + \Delta t)}{N_{\text{neuron}}} \tag{3-5}$$

One problem that arises when using rate coding schemes is the need for multiple spikes to approximate a single floating point number. However, the energy consumption of an SNN increases with the number of spikes used. Therefore, we should minimize the number of spikes required in order to achieve an efficient method of transmitting information.[104] Encoding schemes that consider the exact timing of when the spikes occur can significantly reduce the number of spikes. Auge et al.[105] subdivide temporal coding schemes into temporal contrast, correlation and synchrony, latency or inter-spike-interval (ISI), global referenced, and filter and optimizer encoding. An overview of the different subcategories can be seen in figure 3-2. In the following we focus on global reference and latency-based coding schemes, as these encoding schemes can represent floating point numbers by a single spike of a neuron. Correlation and synchrony coding transform the input data into a spatio-temporal spike representation. This higher-dimensional approach relies on the temporal difference to other spiking neurons in the network. Filter and optimizer-based

[102] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 27.

[103] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 28-29.

[104] Davies, M. et al.: Loihi: A neuromorphic manycore processor with on-chip learning (2018).

[105] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021).

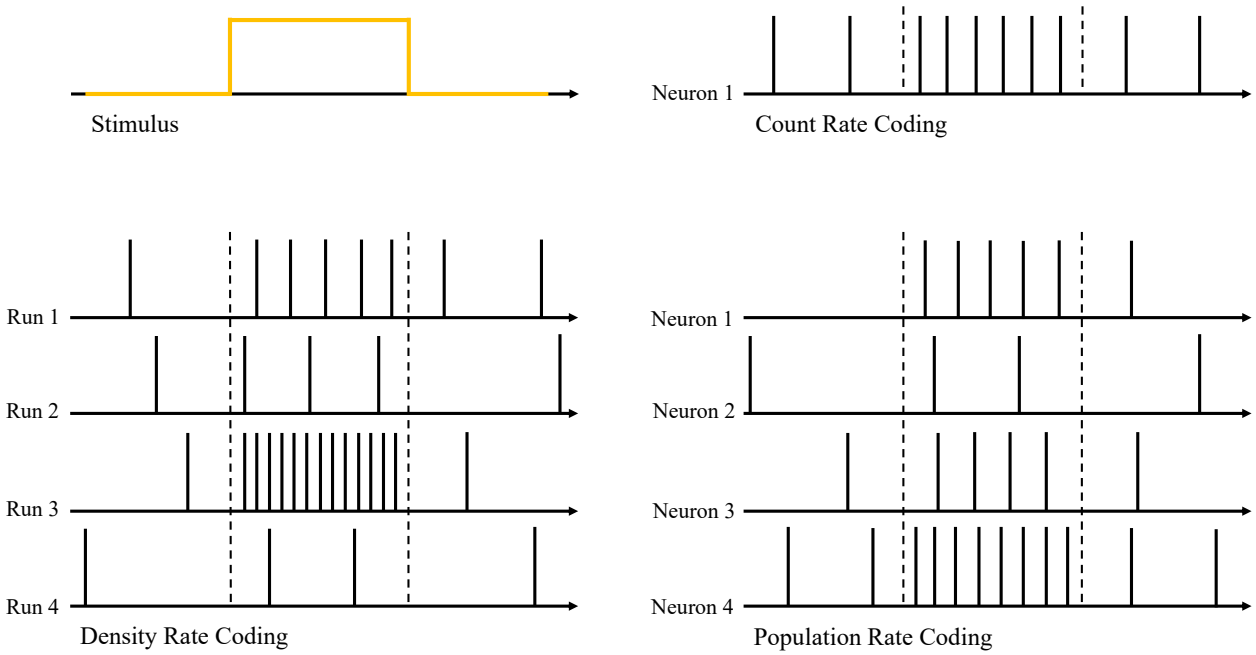

Figure 3-3: Visualization of rate coding schemes.[106]

approaches utilize temporal data streams for data encoding. Since we are trying to encode the ToF of the laser beams and thus do not have a continuous signal, this type of encoding is not particularly suited for our purposes. Temporal contrast coding converts signals into spike trains by observing the change in the input data. As lidar point cloud data are time-invariant, this coding scheme is not applicable.

We already discussed the time-to-first spike (TTFS) encoding scheme. The TTFS encoding is the most basic temporal encoding which utilizes a global time reference. The information is represented by the time difference Δt between the onset of the stimulus and the response of the spiking neuron. The exact firing time can be defined as the inverse of the input amplitude $\Delta t = 1/a$ or a linear relation $\Delta t = 1 - a$, with a as the signal's amplitude. With both definitions, a higher amplitude of the input signal leads to early spikes and vice versa.[106] TTFS encoding was discovered in biological sensory neurons by Johansson and Birznieks. They found that the relative time of the first spike contains information about the direction, force and shape of a surface touched by a finger. One indication of temporal coding in sensory neurons was that tactile information is processed faster than can be easily explained by rate codes.[107] Because temporal coding schemes do not need to calculate an average spike rate when converting floating point numbers, information can be transmitted with less delay.

TTFS coding uses the single reference often the onset of the stimulus to measure the time difference. Instead, phase coding utilizes the relative time difference between the spikes and a global oscillation reference to determine the firing times. In phase coding, the temporal information is described by the relative phase shift between the spikes. Neurons that are more strongly excited by the applied stimulus fire with a short time interval to the amplitude of the oscillation frequency. In addition, several spikes can

[106] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021)

[107] Johansson, R. S.; Birznieks, I.: First spikes in ensembles of human tactile afferents (2004).

occur per oscillation period, which means that the relative time interval between the spikes of a single neuron also carries information. Phase coding has been observed, for example, in the olfactory system of mammals during the recognition of odors.[108]

We can also encode information using the firing order of a population of neurons in relation to a global reference. This approach is applied with rank-order coding (ROC) where the data is encoded without consideration of the exact timing of the spikes. ROC can be described as a discrete normalization filter. During this filtering process the absolute amplitude information is lost. Consequently the absolute signal amplitude or an exactly constant signal cannot be reconstructed based on the rank-order of the spikes.[109] On the other hand, a code based on the firing order is more robust to noisy temporal jitter of each spike than a pure temporal code that must rely on temporal precision.[110] However, a spike sequence can be misinterpreted, if spikes follow each other in quick succession. More sophisticated versions of ROC also consider the ISIs to encode additional information about the spike sequence, so that the order of the spikes can be determined with greater certainty.[111]

Another subcategory of globally referenced encoding schemes are sequential or binary codes. Binary coding translates the information into spikes corresponding to bits with a value of 0 or 1. When we consider a global time reference, there are basically two possibilities to encode the information into bits. In the first approach a spike is interpreted as 1 if it occurs within a given time interval. Otherwise the time interval is assigned to 0.[112] The second option considers the exact timing of the spikes within the time interval. Depending if the the spike occurs in the first or second half of the time interval the spike is encodes as 0 or 1. This guarantees the presence of a spike within each time interval, regardless of the bit value.[113]

In a global referenced coding scheme, the most significant element of a pattern is represented by the first spike of a neuron. This allows a network to predict the output sequence without having processed the entire input sequence. This behaviour can be used to adjust the threshold of neurons to balance the speed and accuracy of the network.[109]

The last coding scheme we will discuss is ISI or latency coding. An ISI is described by the relative time difference between successive spikes. ISI or latency coding uses this relative time difference between the spikes of multiple neurons as a temporal reference, rather than a global time reference.[114] The correlation between the ISI and the intensity of the stimulus was verified in the pyramidal cells of the brown ghost knifefish. These neurons respond to dynamic electrosensory stimuli with bursts and isolated spikes.[115] A subcategory of ISI or latency coding is burst coding. This coding scheme converts the input of the neurons into various inter-spike latencies, where a burst is represented by a several spikes that fall into small ISI.[116]

[108] Hopfield, J. J.: Pattern recognition computation using action potential timing for stimulus representation (1995).

[109] Thorpe, S.; Gautrais, J.: Rank order coding (1998)

[110] Gautrais, J.; Thorpe, S.: Rate coding versus temporal order coding: a theoretical approach (1998).

[111] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021).

[112] Zhang, M. et al.: Efficient spiking neural networks with logarithmic temporal coding (2020).

[113] Hamanaka, H. et al.: Quantized spiking neuron with A/D conversion functions (2006).

[114] Portelli, G. et al.: Rank order coding: a retinal information decoding strategy (2016).

[115] Oswald, A.-M. M. et al.: Interval coding. I. Burst interspike intervals as indicators of stimulus intensity (2007).

[116] Park, S. et al.: Fast and efficient information transmission with burst spikes in deep spiking neural networks (2019).

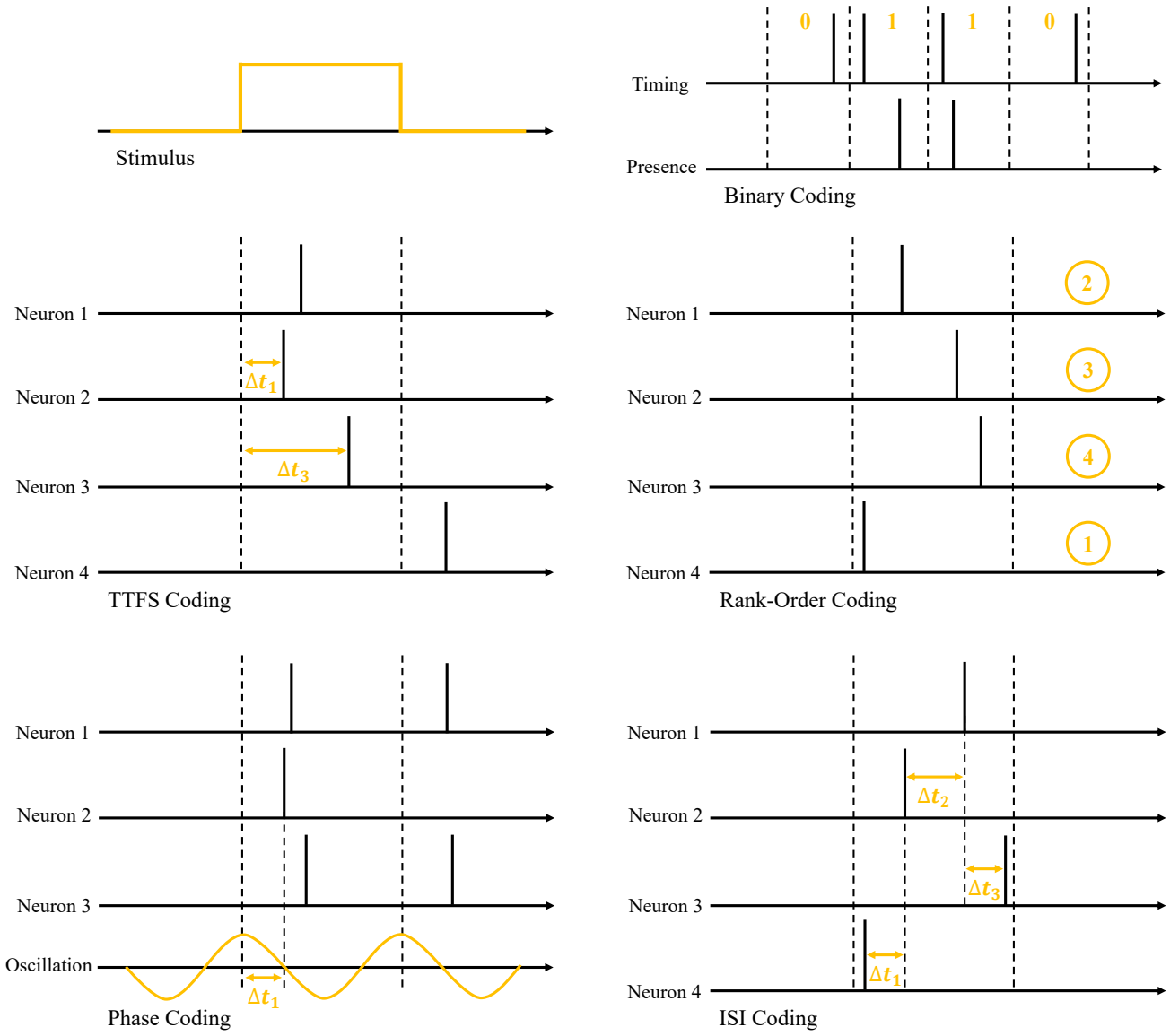

Figure 3-4: Visualization of temporal coding schemes.[117]

Whether a spike belongs to a burst depends on the chosen ISI threshold and the expected number of spikes.[118] An overview of the neural response to an exitory wide pulse using the temporal coding techniques discussed is shown in figure 3-4.

3.3 Point Cloud Object Detection

Lidar sensors output point clouds which are a set of vectors in three dimensional coordinate system. These vectors typically describe the outer surface shape of the objects contained in the FoV. As objects can occur in various positions the points are a unordered collection of data. In addition, not all laser beams are reflected, which means that the number of points also varies. Therefore it is mandatory to preprocess the point cloud data in order to apply deep learning methods which rely on a uniform structure of the

[117] Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021)

[118] Zeldenrust, F. et al.: Neural coding with bursts—current state and future perspectives (2018).

input data.[119] The task of point cloud object detection is to recognize and classify objects such as vehicles, pedestrians, buildings, trees or other elements in point clouds as a basis for decision-making. Current point cloud object detection methods can be divided in projection based, volumetric based and point based approaches.[120] The first section presents benchmark datasets and performance metrics used to train and evaluate 3D object detectors. The next sections review models capable of performing object detection based on lidar data. Special attention is given to those methods that allow the point cloud to be used as input for neural networks. The focus is on the generation of features that are used to detect and classify objects in the point cloud.

3.3.1 Benchmark Datasets

As deep learning approaches rely on training data there is a urgent need for sophisticated and large datasets. Like ImageNet was foundational for the fast development of image classification and 2D object detection networks, the same is needed for three dimensional object detection for automated driving to cover a broad amount of driving scenarios. Large labeled datasets are a key factor for high performing detection networks.[120]

One of the most popular datasets for automated driving is KITTI.[121] This benchmark dataset provides synchronized sensor data for stereo color images, GPS coordinates and also lidar point clouds. Beside motorway scenarios the KITTI dataset includes complex urban areas and narrow countryside roads, which are more challenging for automated vehicles. Furthermore the dataset covers several tasks for automated driving such as stereo matching, visual odometry, 3D tracking and 3D object detection. The lidar dataset contains over 7000 training and test frames, where the objects are labeled by their class and bounding boxes. Depending on size, occlusion and truncation level the objects are categorized in easy, moderate and hard to detect. Although KITTI is widely used, the dataset has certain limitations. One of them is the limited configuration of sensors, as all driving scenes were obtained by the same set of senors. Additionally, all scenes were filmed during daytime and predominantly under sunny conditions. A further weakness of the dataset is the highly unbalanced distribution of the object classes. KITTI is divided into 75% cars, 15% pedestrians and 4% cyclists.

The performance of detection networks can be measured by the mean average precision (mAP). As the KITTI dataset contains multiple classes the mAP is obtained by calculating the mean of the average precision (AP) of the individual classes. A predicted bounding box is considered as a true positive (TP) if the intersection over union (IoU) with the ground-truth bounding box is greater than a given threshold. Otherwise the prediction is marked as false positive (FP). If a labeled object is not recognized, it is referred to as a false negative (FN). These classification errors are used to calculate precision and recall metrics like in equation 3-6 and 3-7. Based on predicted and ground-truth bounding boxes we get a recall distribution $p(r)$ with which the AP can be calculated as in equation 3-8.

[119] Liang, W. et al.: A survey of 3D object detection (2021).

[120] Arnold, E. et al.: A survey on 3d object detection methods for autonomous driving applications (2019)

[121] Geiger, A. et al.: Are we ready for autonomous driving? the kitti vision benchmark suite (2012).

$$precision = \frac{TP}{TP + FP} \tag{3-6}$$

$$recall = \frac{TP}{TP + FN} \tag{3-7}$$

$$AP = \int_0^1 p(r)dr \tag{3-8}$$

As many ADS use multi-modal detection methods based on 360 degree recordings the KITTI development team proposed the KITTI360 dataset.[122] This dataset first annotates three-dimensional scene elements and then transfer this information into the image domain. In this manner KITTI360 provides a dense semantic instance annotation for 3D point clouds and 2D images. However, the lack of varied driving scenarios and the relatively low number of frames by today's standards limit the reliability of detection networks in real-world applications.

A dataset that overcomes some of this limitations is the nuScenes dataset[123] developed by Motional. It is one of the largest labeled datasets specifically for automated driving with 360 degree coverage of all vision and distance sensor modalities, with a total of 1000 driving scenes of 20 seconds each, divided into 700 training scenes, 150 validation scenes and 150 test scenes. All scenes are fully annotated with 3D bounding boxes for 23 classes and 8 attributes. The dataset was collected by two vehicles equipped with 5 radar sensors, 6 cameras and a lidar and has 7 times more annotations and 100 times more images than the KITTI dataset. The vehicles drove in Boston and Singapore, two cities with dense traffic and challenging driving situations. The dataset covers urban, residential, natural and industrial locations with sunny, rainy and cloudy weather conditions. In addition, nuScenes includes right-hand and left-hand traffic. Unlike the KITTI dataset, the calculation of the AP for each class is not overly based on the traditional bounding box. Instead, nuScenes uses a center distance based metric and considers the average of the TPs in terms of translation, scale, orientation, speed and attribute error, resulting in the nuScenes detection score (NDS), which can be calculated using the equation 3-9. This metric provides a more comprehensive description of detection performance than the mAP.

$$NDS = \frac{1}{10}[5mAP + \sum_{mTP \epsilon TP} (1 - min(1, mTP))] \tag{3-9}$$

Another high-quality annotated automated driving dataset is Waymo Open. This dataset is collected by Waymo's own fleet of self-driving vehicles. Like the nuScenes dataset, Waymo Open contains 1000 scenes, divided into 798 training scenes and 202 validation scenes, with 2D and 3D annotated labels. A key difference is the extensive use of lidar sensors. In total, the dataset is captured by 5 lidar and 5 high-resolution cameras, each capturing 20 seconds of continuous driving. Like the KITTI dataset,

[122] Xie, J. et al.: Semantic instance annotation of street scenes by 3d to 2d label transfer (2016).

[123] Caesar, H. et al.: nuscenes: A multimodal dataset for autonomous driving (2020).

Waymo Open relies on AP as its main metric, but also proposes a new metric that takes into account the heading accuracy of the predicted bounding box.[124]

As the generation of such large datasets is a time-consuming and expensive process, efforts have been made to overcome the limitations of real-world recordings by using virtual environments. The virtual KITTI dataset[125] is a recreation of the real KITTI dataset using a game engine. However, the virtual environment can be manipulated to create more varied driving situations with different weather conditions or sensor positions. The objects are also automatically annotated, which is usually the most time-consuming step. Virtual KITTI provides nearly 17000 frames that can be transferred to the real KITTI dataset. This makes it possible to pre-train an object detection network with virtual data and then fine-tune the model with the real dataset. Models trained only with virtual KITTI have a minimal performance gap compared to models trained with the original KITTI dataset. The use of virtual environments for training automated driving systems opens up new possibilities for making object detection more reliable. Another open source simulation framework for automated driving is CARLA.[126] This simulation tool provides two virtual cities where different 3D models for cars and pedestrians can be selected. As with the Virtual KITTI dataset, weather and lighting conditions can be adjusted to create more challenging driving situations.

3.3.2 Projection Methods

Object detection for 2D images is a well-researched topic in computer vision. There are many datasets and architectures available for this type of data, leading to the idea of applying them to three-dimensional point clouds. Therefore, point cloud projection methods are an established approach that transform the point cloud into a 2D image in order to use common 2D object detection frameworks. The point cloud data can be transformed into a 2D image using either a plane, cylindrical or spherical projection. The three dimensional bounding boxes can be determined using position and dimensional regression.

VeloFCN[127] is a detection model that uses a cylindrical projection of the point cloud as input data for a CNN to predict 3D bounding boxes of vehicles. The images resulting from the projection carry height and distance information from the original lidar points and are down-sampled in three steps using a 2D CNN. The final layer has two outputs. The first output determines whether a point is part of a vehicle or the background. The second output estimates the 3D bounding boxes. As the model produces multiple proposals for the bounding boxes, non-maximum suppression is used to obtain unique bounding boxes. The detection model has been trained end-to-end on the KITTI dataset. However, VeloFCN can only recognize vehicles and has difficulty detecting small or obscured objects that are only represented by a few points.

Another option is to map the point cloud onto a 2D bird's eye image. BirdNet[128] is a 3D object detection model that projects the lidar point cloud into cells containing height, intensity and density information from a bird's eye view. As the projection is similar to a three-channel RGB image, BirdNet uses Faster

[124] Sun, P. et al.: Scalability in perception for autonomous driving: Waymo open dataset (2020).

[125] Gaidon, A. et al.: Virtual worlds as proxy for multi-object tracking analysis (2016).

[126] Dosovitskiy, A. et al.: CARLA: An open urban driving simulator (2017).

[127] Li, B. et al.: Vehicle detection from 3d lidar using fully convolutional network (2016).

[128] Beltrán, J. et al.: Birdnet: a 3d object detection framework from lidar information (2018).

R-CNN[129] to generate region proposals and then classifies these proposals into the different classes. A novelty of BirdNet compared to other approaches using the same projection strategy is the normalization of the point density channel. This is particularly helpful as data collected by lidar, especially with high point density, leads to a large variation in the density channel. Furthermore, the normalization of the density channel makes the feature map insensitive to the specifications of the lidar. Despite the decent performance of bird's eye models, they suffer from poor orientation angle regression.

In addition to high predictive performance, real-time operation is a fundamental requirement for perception networks used in automated driving. To address this challenge, Complex-YOLO[130] is a single-shot detector based on the YOLO9000 architecture[131], which focuses on efficiency and detection speed. This model extends YOLO to predict an extra dimension and yaw angles. Complex-YOLO achieves a runtime of 50 frames per second (FPS), which is up to five times faster than previous methods. However, there is a trade-off between inference time and detection performance. Complex YOLO also underperforms in terms of detection accuracy compared to networks using region proposal networks.

Besides detection speed and accuracy, the confidence of a prediction made by a neural network is a key parameter to ensure safe operation of the vehicle. Most detection frameworks use soft-max normalization to obtain the likelihoods for the different classes. However, the likelihoods do not necessarily indicates the absolute confidence in the prediction. TowardsSafe[132] addresses this issue by using a Bayesian neural network to predict the class and 3D bounding box after ROI pooling, which allows to quantify the confidence of the model on both outputs. As the estimation of the uncertainty requires several forward passes of the network, this method limits real-time object detection.

A limitation common to all projection-based methods is the inevitable loss of information when mapping the point cloud to a two-dimensional image. Also, an end-to-end learned representation of the input is preferable to hand-crafted features such as point density, as these can cause an unintentional bias.

3.3.3 Volumetric Methods

Volumetric approaches discretise the point cloud into a three-dimensional grid where each element is called a voxel. Each unit carries attributes derived from the points in the voxel, such as binary occupancy or a continuous point density. This unified representation is then processed by 3D convolutional networks.

3DFCN[133] is a further development of VeloFCN that uses a binary volumetric input to detect vehicles. As in the previous work, the model predicts two outputs, the first being the estimated region of interest and the second a prediction of the object coordinates. A challenge faced by all methods that use a volumetric representation of the point cloud is that most voxels are empty due to the high sparsity of the point cloud. In addition, three-dimensional convolution drastically increases the computational cost, resulting in a high inference time of 1 FPS for 3DFCN.

[129] Ren, S. et al.: Faster r-cnn: Towards real-time object detection with region proposal networks (2015).

[130] Simon, M. et al.: Complex-yolo: Real-time 3d object detection on point clouds (2018).

[131] Redmon, J.; Farhadi, A.: YOLO9000: better, faster, stronger (2017).

[132] Feng, D. et al.: Towards safe autonomous driving (2018).

[133] Li, B.: 3d fully convolutional network for vehicle detection in point cloud (2017).

To address some of these challenges, Vote3Deep[134] is a 3D object detector that proposes a more efficient convolution that applies the convolution filter only to non-zero cells. The inherent sparsity in the input data is exploited by using a voting algorithm, where each non-zero feature vector casts a set of votes for its surrounding cell in the output layer. These voting weights are obtained by inverting the convolutional filter along each spatial dimension. To maintain sparsity throughout the model, the non-linear ReLU activation function and L1 regularization are used. Unlike 3DFCN, Vote3Deep is able to predict vehicles, cyclists and pedestrians by running independent models for each class, but also sacrifices the inference time gain of sparse convolution. In addition to the detection performance of volumetric methods, they lag in real-time applicability, which limits their use in automated driving.

3.3.4 Point Based Methods

Since point clouds consist of sparsely distributed 3D points in space, it is not obvious to incorporate their structure into neural networks that assume a fixed input data size. The approaches discussed so far either transform point clouds into images using different projection methods or into a volumetric representation such as voxels. The third way to deal with the irregularity of point clouds is to apply a symmetric function to obtain a pointwise representation that is invariant to the input order.

PointNet[135] was the first neural network to propose this method. This model uses segmented 3D point clouds as input to perform object classification and part segmentation. To use the raw point cloud data, the network applies a point-by-point transformation using fully connected layers and aggregates the output into a global feature vector using a max-pooling layer. Before passing the data to the multi-layer perceptrons, a transformation is performed by so-called T-Nets. These networks estimate a transformation matrix to rotate the point cloud, which helps the network to recognize the given objects. The resulting global feature is invariant to the order of the input points and allows PointNet to process the raw point cloud data directly. Most objects can be described by a few critical points. PointNet is able to represent these meaningful points in the global feature, which makes the network extremely robust to various types of input corruption. Even if 50% of the points are missing by chance, the recognition model is still able to predict the class of the object with a performance drop of less than 4%.

The ability of PointNet to learn features directly from point cloud data was further enhanced with PointNet++[136]. It introduced a hierarchical neural network that recursively applies the feature generation of PointNet to a nested partition of the input point cloud. This approach allows PointNet++ to learn additional local features to detect fine-grained patterns and better generalize to complex scenes.

PointNet is limited to segmented point clouds containing only a single object. VoxelNet[137] closes the gap to object classification and detection usable for automated driving. This network combines the point-wise feature generation of PointNet with volumetric representation methods to obtain voxel-wise features. First, a fixed number of points are randomly selected to reduce the computational complexity and improve the generalization of the model. Next, each set of points is processed through the voxel-feature-encoder layer to obtain a four-dimensional representation of the point cloud. The feature learning network is followed

[134] Engelcke, M. et al.: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks (2017).

[135] Qi, C. R. et al.: Pointnet: Deep learning on point sets for 3d classification and segmentation (2017).

[136] Qi, C. R. et al.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space (2017).

[137] Zhou, Y.; Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection (2018).

by 3D convolutional layers and a region proposal network to predict the bounding boxes and classes. As with the volumetric methods discussed previously, VoxelNet's spatial discretisation produces many empty grid cells, resulting in unnecessary computational overhead.

To increase the speed of training and inference, a model called SECOND[138] added an improved sparse convolution method. This network gathers all non-zero voxels before the convolutional layer and scatters the result to the original dimensions. In addition, SECOND proposes a novel angle loss regression approach that improves the orientation regression performance. SECOND achieves an inference speed of 20 FPS on the KITTI dataset, more than four times faster than VoxelNet.

To achieve the goal of real-time inference, PointPillars[139] proposes a novel point cloud encoder that converts the lidar data into a two-dimensional pseudo-image. The encoded features can be processed by any standard 2D convolutional detection framework. To obtain the pseudo-image, the point cloud is first discretised into an evenly spaced grid in the xy-plane to generate pillars, which are like voxels with unlimited spatial extent in the z-dimension. The points within a pillar are then decoded using a simplified version of the PointNet feature generation. PointPillars uses a single shot detector to perform 3D object detection and achieves an inference speed of 62 FPS on the KITTI dataset.

3.4 Spiking Neuron Models

SNNs have been described as the third generation of neural networks. They attempt to bridge the gap between machine learning and neuroscience.[140] As we discussed in the previous chapters, ANNs and SNNs rely on fundamentally different neuron models. Where ANNs typically use computational units such as sigmoid, tanh or ReLU, SNNs use non-differentiable neuron models, which we will now investigate in more detail. The different neuron models can be distinguished by their biological plausibility and their computational complexity, which we will measure in floating point operations (FLOPS) required to simulate the neuron model over a single time step. In addition to the SRM described in chapter 2.3, we review three other popular neuronal models, including the Hodgkin-Huxley model, the Izhikevich model, and the integrate-and-fire model.

3.4.1 Hodgkin-Huxley Model

The Hodgkin-Huxley model was the first spiking neuron model to describe the initiation and propagation of action potentials.[141] If we think of a neuron as an electric circuit, an electric current can flow across the cell membrane either by charging the membrane capacitor or by ions flowing across the resistor in parallel with the capacitor. Hodgkin and Huxley studied the flow of electric current through the surface membrane of a squid giant axon. They observed three different ionic currents consisting of sodium Na and potassium K ions and a small leakage current L mainly made of chloride ions.[142] The total ionic current I_i is the sum of the ionic currents for a single ion channel I_k involved and can be calculated using the equations 3-10 and 3-11, where E is the equilibrium potential and u is the membrane potential. The

[138] Yan, Y. et al.: Second: Sparsely embedded convolutional detection (2018).

[139] Lang, A. H. et al.: Pointpillars: Fast encoders for object detection from point clouds (2019).

[140] Maass, W.: Networks of spiking neurons: The third generation of neural network models (1997).

[141] Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022).

[142] Hodgkin, A. L.; Huxley, A. F.: A quantitative description of membrane current (1952).

so-called gate equations 3-12 and 3-13 control the flow of sodium ions. The potassium ions are controlled by the gate equation 3-14. The transition rate of each gate is determined by the variables α and β, which range from non-permissive to permissive and vice versa.[143]

$$I_i = \sum_k I_k = \sum_k G_k(u - E_k) \tag{3-10}$$

$$I_i = G_{Na} m^3 h(u - E_{Na}) + G_K n^4 (u - E_K) + G_L(u - E_L) \tag{3-11}$$

If an external stimulus raises the membrane potential, the conductance G of the sodium channels increases, resulting in a positive flow of sodium ions into the soma, and the membrane potential rises even further. If this positive feedback is strong enough, an action potential will propagate along the axon. This event is followed by a period of relative refractoriness during which a second spike is possible but requires a stronger stimulus. The Hodgkin-Huxley model allows this phenomenon to be studied in more detail. Two factors are responsible for the refractoriness of the neuron. First, the refractory membrane potential is even lower than the resting potential of the neuron due to the hyperpolarising spike afterpotential. Second, because many ion channels are still open immediately after the action potential is triggered, the membrane resistance is lower than when the neuron is at rest. Therefore, the stimulating current decays faster due to this depolarizing effect. Both of these factors result in a larger stimulus being needed to reach the threshold shortly after a spike has been emitted.[144]

$$\frac{m}{dt} = \alpha_m(u)(1 - m) - \beta_m(u)m \tag{3-12}$$

$$\frac{h}{dt} = \alpha_h(u)(1 - h) - \beta_h(u)h \tag{3-13}$$

$$\frac{n}{dt} = \alpha_n(u)(1 - n) - \beta_n(u)n \tag{3-14}$$

The equations of the Hodgkin-Huxley model provide a comprehensive description of the electrophysiological properties of the squid giant axon. The model is able to simulate spike generation based on sodium and potassium ions. Basically, an action potential is generated by a brief influx of sodium ions followed by an efflux of potassium ions. This principle is also found in higher organisms. However, cortical nerve cells in vertebrates such as humans have more complex electrophysiological properties than the squid axon.[145]

High-dimensional non-linear differential equations are difficult to solve. Therefore, a reduction of the four-dimensional Hodgkin-Huxley model to a two-dimensional model is desirable, since two-dimensional

[143] Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019), pp. 137-138.

[144] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 44-46.

[145] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 51.

differential equations can be analyzed using phase planes. A general approach is to consider the gating variable m as an instantaneous variable, since its dynamics is much faster than that of the variables n, h and u. This is known as a quasi-steady-state approximation. To further reduce the dimensionality, the variables n and $(1 - h)$ are approximated by a single effective variable w. The three gate equations resolve into a single equation as the m equation disappears since m is treated as instantaneous and the two equations for n and h are expressed by a single effective equation for the variable w.[146] Fritzhugh and Nagumo were among the first to reduce the four equations of the Hodgkin-Huxley model to two equations.[147] A further reduction of the Hodgkin-Huxley model was proposed by Morris and Lecar.[148]

The computational complexity of the Hodgkin-Huxley model makes it less suitable for deep neural network applications.[149] The model requires approximately 1200 FLOPS to simulate the neural dynamics for a single time step. Even the simplified version of the Hodgkin-Huxley model using two of the four equations still requires several hundred FLOPS.[150] Nevertheless, the Hodgkin-Huxley model is the basis of many other simplified spiking neuron models that describe the dynamics of the neuron by modeling the flow of ions.[151]

3.4.2 Integrate-and-Fire Model

Conductance-based neuron models, such as the Hodgkin-Huxley model, are capable of simulating the electrophysiological behavior of biological neurons. However, their intrinsic complexity makes them difficult to analyze and computationally expensive. On the other hand, phenomenological models of biological neurons, such as the integrate-and-fire model, are well suited to describe basic neural dynamics. These threshold models generate spikes whenever the membrane potential u exceeds a certain limit ϑ.

The integrate-and-fire neuron can be modeled by an electrical circuit consisting of a capacitor C in parallel with a resistor R to which a current $I(t)$ is applied. The driving current $I(t)$, which consists of the resistive current I_R and the capacitive current I_C as in equation 3-15, can be either an external current or a presynaptic stimulus. The resistance R and the capacitance C are combined by the time-independent parameter τ_m, resulting in the equation 3-16, which is the standard form of the integrate-and-fire neuron. The variable u refers to the membrane potential and the parameter τ_m is the membrane time constant of the neuron. In contrast to conductance-based neuron models, the shape of a spike is not described by the equations. According to the integrate-and-fire model, an action potential is just an event that occurs at the firing time $t^{(f)}$ when the membrane potential u reaches the threshold ϑ. After the firing time $t^{(f)}$, the membrane potential u is immediately reset to the resting potential u_{res}.[152] The leaky integration of equation 3-16 and the resetting of the membrane potential after a spike is the basic integrate-and-fire model proposed by Stein in 1967.[153] Since biological neurons go through a period of refractoriness, integration

[146] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 77-79.

[147] FitzHugh, R.: Impulses and physiological states in theoretical models of nerve membrane (1961).

[148] Morris, C.; Lecar, H.: Voltage oscillations in the barnacle giant muscle fiber (1981).

[149] Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022).

[150] Paugam-Moisy, H.; Bohte, S. M.: Computing with spiking neuron networks. (2012).

[151] Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019), p. 138.

[152] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), pp. 101-103.

[153] Stein, R. B.: Some models of neuronal variability (1967).

can be interrupted after the release of an action potential by defining an absolute refractory time Δ^{abs}.[154]

In fact, the leaky integrate-and-fire model is a special case of the spike response model discussed in chapter 2.3.2. The behavior of the leaky integrate-and-fire model, where the membrane potential u evolves over time as the model integrates incoming synaptic currents $I(t)$, can be defined within the framework of the spike response model by choosing appropriate synaptic response kernels ϵ_{ij}. The leakage of the membrane potential u, which is essentially a low-pass filtering of the membrane potential itself, can be simulated with the spike response model by incorporating self-inhibition terms.[155]

$$I(t) = \frac{u(t)}{R} + C\frac{du}{dt} \tag{3-15}$$

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{3-16}$$

The integrate-and-fire model is very popular as it requires only 5 FLOPS to simulate the neural dynamics over one time step. Other variants of the integrate-and-fire model, such as the integrate-and-fire with adaptation and the quadratic or exponential integrate-and-fire model, require about 10 FLOPS per millisecond.[156] In addition, the leaky integrate-and-fire model can be implemented on analogue hardware as the integration and leakage of the membrane potential can be modelled by sub-threshold transistors and capacitors.[157]

3.4.3 Izhikevich Model

The Izhikevich model offers a compromise between biological plausibility and computational complexity. The model consists of a two-dimensional system of ordinary differential equations 3-17 and 3-18 with the auxiliary after-spike reset 3-19. The Izhikevich model is represented by the dimensionless variables q and u and the parameters a, b, c and d. The membrane potential of the neuron is expressed by the variable u. The variable q describes the membrane recovery as a result of the activation of the potassium ionic current and the inactivation of the sodium ionic current. The membrane voltage and the recovery variable are reset as described in the equation 3-19 after the action potential has reached its maximum. Depending on the choice of parameters a, b, c and d, different fire patterns found in biological neurons can be simulated. The a parameter determines the rate at which the neuron recovers from a spike. Higher values result in a faster recovery and therefore a shorter refractory period. Sub-threshold fluctuations are accounted for by the parameter b. Higher values of this parameter result in a stronger coupling of the variables u and v. The resting potential of the neuron is between -70 mV and -60 mV depending on the value of the parameter b. The parameter c describes the reset value of the membrane potential u after a spike has been emitted by

[154] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 103.

[155] Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002), p. 116.

[156] Paugam-Moisy, H.; Bohte, S. M.: Computing with spiking neuron networks. (2012).

[157] Aamir, S. A. et al.: An accelerated LIF neuronal network array (2018).

the neuron and is typically around -65 mV. Similarly to the c parameter, the d parameter determines the post-spike reset of the recovery variable q, typically this value is 2.[158]

$$\frac{du(t)}{dt} = 0.04u^2 + 5u + 140 - q + I(t) \tag{3-17}$$

$$\frac{dq(t)}{dt} = a(bu - q) \tag{3-18}$$

$$\text{if } u \geq 30 \text{ mV, then } \begin{cases} u = c \\ q = q + d \end{cases} \tag{3-19}$$

Unlike simpler models, the Izhikevich model is able to simulate different neuron types found in the mammalian brain, including regular spiking, intrinsically bursting, chattering, fast spiking, low-threshold spiking, thalamo-cortical and resonator neurons. In addition to the wide range of neuron types that can be mapped, the model is also capable of building large networks consisting of thousands of artificial spiking neurons[158], as it requires only about 13 FLOPS to simulate neural dynamics over a period of 1 ms.[159] The Izhikevich model offers a trade-off between biological plausibility and computational complexity. Due to its relatively low computational requirements, the model is suitable for building SNNs for engineering problems.[160]

3.5 SNN Learning Rules

Training SNNs presents unique challenges due to their discrete and time-dependent nature.[161] One of the main challenges is the lack of a widely adopted standardised training algorithm, such as the back-propagation used in traditional ANNs.[162] There are three general approaches to SNN learning. The first is unsupervised learning such as spike timing dependent plasticity (STDP). The second is indirect unsupervised learning via ANN-to-SNN conversion, and the third is direct supervised learning, which attempts to apply gradient descent-based back-propagation.[163] Each approach has its own unique challenges and advantages, which are discussed in the following sections.

3.5.1 Unsupervised Learning

Unsupervised learning is a field of machine learning that aims to detect patterns and structures in data without prior labeling or categorization. For SNNs, this machine learning paradigm is based on the Hebbian rule, which consists of adapting the network's synaptic connections to the data received by

[158] Izhikevich, E. M.: Simple model of spiking neurons (2003)

[159] Paugam-Moisy, H.; Bohte, S. M.: Computing with spiking neuron networks. (2012).

[160] Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022).

[161] Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019), p. 146.

[162] Pfeiffer, M.; Pfeil, T.: Deep learning with spiking neurons: Opportunities and challenges (2018).

[163] Wu, Y. et al.: Direct training for spiking neural networks: Faster, larger, better (2019).

the neurons.[164] A common learning paradigm inspired by the Hebbian rule is spike-timing-dependent plasticity (STDP).[165] The basic idea behind STDP is that the effectiveness of a synaptic connection between two neurons depends on the timing of their action potentials. If a presynaptic neuron fires an action potential just before a postsynaptic neuron, the synaptic connection is strengthened. In other words, if the presynaptic neuron helps to excite the postsynaptic neuron, the synapse is strengthened, and this is called long-term potentiation (LTP). Conversely, if the presynaptic neuron fires an action potential shortly after the postsynaptic neuron, the synaptic connection is weakened. When the presynaptic neuron contributes less to the excitation of the postsynaptic neuron, this is called long-term depression (LTD). The ability of neurons to detect and respond to the timing of their action potentials is critical for encoding information in the brain. STDP allows the brain to adapt to repeated patterns of activity, ultimately leading to neural circuits optimized for specific learning and memory tasks. The most common STDP rule is described by the equation 3-20.

$$F(\Delta t) = \begin{cases} A_+ \exp(\Delta t/\tau_+) & \text{if } \Delta t < 0 \\ -A_- \exp(\Delta t/\tau_-) & \text{if } \Delta t \geq 0 \end{cases} \tag{3-20}$$

The function $F(\Delta t)$ determines the amount of synaptic modification that results from a single pair of pre- and postsynaptic actions potentials separated by the time Δt. The parameters τ_+ and τ_+ define the ranges of pre- and postsynaptic ISIs over which either synaptic strengthening or weakening occurs. The parameters A_+ and A_- are always positive and determine the maximum amount of synaptic modification, when the time difference Δt between pre- and postsynaptic spike is close to zero.

The synaptic weights can be updated using the learning algorithm from the generalized Widrow-Hoff learning rule, also known as the delta rule.[166] Similar to other training algorithms, the synaptic weights of a neuron are iteratively adjusted using the equation 3-21 to achieve the desired output spike sequence for the given input data. The change in synaptic weights Δw is obtained by multiplying the result of the learning rule $F(\Delta t)$ by a positive learning rate λ. Depending on the value of the learning rate the synaptic weights change faster or slower.

$$\Delta w = \lambda F(\Delta t) \tag{3-21}$$

Through STDP, connected neurons are able to learn successive temporal associations from data and form chains of connections to represent the patterns in the data. Masquelier et al. have shown that a single leaky integrate-and-fire neuron can be trained to respond quickly to the onset of a spatio-temporal spiking pattern at 2000 synapses to which the neuron has previously been presented, even if 1000 of them induce noise. The more times the neuron has been presented with the pattern, the faster it detects the onset of the pattern.[167] Although STDP is biologically plausible, it only takes into account local information

[164] Hebb, D. O.: The organization of behavior: A neuropsychological theory (2005).

[165] Song, S. et al.: Competitive Hebbian learning through spike-timing-dependent synaptic plasticity (2000).

[166] Anderson, J. A.: A simple neural network generating an interactive memory (1972).

[167] Masquelier, T. et al.: Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains (2008).

transmitted between individual neurons. The lack of global information hinders the convergence of large SNNs, especially when they are confronted with more complex datasets.[168]

3.5.2 Supervised Learning

One of the first learning algorithms that could be used to train SNNs in a supervised manner was SpikeProp.[169] This algorithm uses the backpropagation of errors to adjust the synaptic weights of the network. In order to apply backpropagation, the derivative of the activation functions must be calculated. ANNs typically use continuous activation functions, such as the sigmoid or ReLU, which allow straightforward derivative calculations. However, since SNNs use discrete action potentials, these derivatives must be approximated. This non-linearity of SNNs complicates the backpropagation of errors through the network. SpikeProp uses so-called surrogate gradients instead of the actual derivatives of the spike functions. Surrogate gradients are smooth and differentiable functions that approximate the non-differentiable spikes. SpikeProp can be used to learn a set of target spiking times of output neurons for a given set of input spikes. During training, the error between the actual output times of the network and the expected output times is calculated. Then the gradients of the error are backpropagated using the surrogate gradients of the spike functions. Bohte et al. applied this learning algorithm successfully to classification problems using a three-layer architecture. They proposed the least mean squares as the error function, but other choices like entropy are also possible. SpikeProp can be applied to both temporal and rate-coded networks.

An advanced version of SpikeProp is Spike Train SpikeProp (ST-SpikeProp).[170] This learning algorithm provides a multi-spike learning algorithm with error backpropagation for single neurons and multi-layer SNNs. ST-SpikeProp improves the learning accuracy compared to the regular SpikeProp algorithm when there is a relatively large number of output spikes to learn. However, SpikeProp and ST-SpikeProp are less suitable for training large networks, as their computational complexity leads to long training times, which becomes problematic as the size of the network increases.[171]

Another method for supervised learning of SNNs is the spatio-temporal backpropagation (STBP) algorithm.[172] This learning rule combines the time-dependent temporal domain and the layer-by-layer spatial domain. An approximate derivative for spike activity, suitable for gradient descent training, is proposed to solve the non-differentiable problem of SNNs. A fully connected and a convolutional SNN trained with STBP achieved state-of-the-art accuracy on MNIST and Neuromorphic-MNIST, a dataset specifically designed for SNNs. Supervised learning using temporal coding has been shown to significantly reduce the energy consumption of SNNs by considering the temporal domain. However, many supervised learning algorithms for SNNs are not as well optimized as for ANNs, making it more difficult to explore deeper networks.[173]

[168] Masquelier, T.; Thorpe, S. J.: Unsupervised learning of visual features through spike timing dependent plasticity (2007).

[169] Bohte, S. M. et al.: Error-backpropagation in temporally encoded networks of spiking neurons (2002).

[170] Xu, Y. et al.: A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks (2013).

[171] Pfeiffer, M.; Pfeil, T.: Deep learning with spiking neurons: Opportunities and challenges (2018).

[172] Wu, Y. et al.: Spatio-temporal backpropagation for training high-performance spiking neural networks (2018).

[173] Wu, Y. et al.: Direct training for spiking neural networks: Faster, larger, better (2019).

3.5.3 ANN-to-SNN Conversion

Converting an already trained ANN into a SNN is another approach to train SNNs. This method allows to access the advantages of well-established ANN models and use them for SNNs. ANN-to-SNN conversion is also referred to as spike transcoding or spike conversion. Converting an ANN into a SNN has many advantages. First, simulating the exact spike dynamics in a large network can be computationally expensive, especially when high firing rates and precise spike times are required. SNN-to-ANN conversion allows SNN to be applied to complex problems that have already been solved or are easier to compute with ANNs. Compared to their formal ANN counterparts, the loss of accuracy is relatively small. Second, ANN-to-SNN conversion allows state-of-the-art training methods developed for ANNs and many modern deep networks to be applied to SNNs. In addition, the time-consuming hyper-parameter tuning for ANNs can be accelerated using well-implemented optimization algorithms on GPUs.[174]

The main challenge of ANN-to-SNN conversion is the representation of negative values and biases in spiking neurons. These problems can be avoided by using ReLU as the activation function, which is always greater than or equal to zero, and by computing without bias values. The typical max-pooling layers that follow the convolutional layers are replaced by an average pooling. Diehl et al. proposed a method for converting fully connected neural networks and CNNs into SNNs with minimal performance loss in the conversion process. They used a novel weight normalization method to regulate the firing rates. To accomplish this, they propagated a subset of training data through the network, observed the firing rates in each layer, and rescaled the input weights to each layer to achieve the desired firing rates.[175] This normalization method was later extended to even larger networks by Rueckauer et al.[176] and Sengupta et al.[177], increasing robustness against outliers and considering actual firing rates in weight normalization.

However, the ANN-to-SNN conversion adds many restrictions to the pre-trained ANN. For example, bias terms cannot be used, pooling is limited to average pooling, and only ReLU can be used as the activation function of the ANN because negative values are difficult to represent using spikes. Also the encoding of SNNs is limited to rate-based methods, since the continuous activation functions of ANNs are commonly approximated by the firing rate of the neurons. This in turn increases the number of spikes required, resulting in a higher energy consumption of the SNN.[178] Furthermore, this indirect supervised learning approach gives little insight into how SNNs work and learn, as they are only used for inference. In addition, deep SNNs require several hundred simulation time steps to achieve satisfactory performance.[179]

3.6 Hardware Implementation for SNNs

The information processing in the human brain differs fundamentally to traditional computing systems. Classic computer hardware, such as CPUs (central processing units) and GPUs (graphics processing units), are designed to perform algorithmic tasks and calculations. This hardware is based on the von Neumann architecture, a fundamental concept in computer technology developed by John von Neumann in the

[174] Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022).

[175] Diehl, P. U. et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing (2015).

[176] Rueckauer, B. et al.: Conversion of continuous-valued deep networks (2017).

[177] Sengupta, A. et al.: Going deeper in spiking neural networks: VGG and residual architectures (2019).

[178] Pfeiffer, M.; Pfeil, T.: Deep learning with spiking neurons: Opportunities and challenges (2018).

[179] Wu, Y. et al.: Direct training for spiking neural networks: Faster, larger, better (2019).

1940s. The architecture subdivides the structure of a computer in a central processing unit, a memory, a bus system and an input/output (I/O) interface. The von Neumann architecture is characterized by the sequential execution of operations, in which the CPU successively retrieves instructions from the memory, interprets them and executes them. This process is known as the von Neumann cycle and involves loading, decoding, executing and saving instructions. A key feature of this architecture is that the memory stores both instructions and data, which allows for high flexibility, but can also lead to bottlenecks in data transfer between CPU and memory, known as the von Neumann bottleneck.[180] Although the von Neumann architecture is suitable for general-purpose and sequential tasks, it is limited when it comes to complex, parallel or time-triggered tasks such as those found in SNNs.

The implementation of SNNs requires special computing hardware to unlock the full potential of this technology. In order to imitate the temporal dynamics of biological neurons, this type of hardware must enable asynchronous communication and massive parallelisation. Conventional computer hardware such as CPUs or GPUs use a global clock to enable synchronized execution of operations, which is one of the reasons for their high power consumption. This high communication overhead not only results in high power consumption, but also slows down the computation for both inference and training of the network. More advanced aspects, such as synaptic plasticity, are also difficult to implement on traditional computing systems. Field programmable gate arrays (FPGAs) and neuromorphic hardware are two chip designs that can overcome the communication bottleneck and accelerate the inference and training of SNNs.[181]

FPGAs are flexible, programmable circuits that enable digital circuit design at the hardware level. Unlike the traditional von Neumann hardware used in most computers, FPGAs allow parallel processing and can be adapted to the specific requirements of different circuits. The architecture of von Neumann computers is based on a central processor that executes instructions sequentially and accesses a shared memory, while FPGAs allow multiple operations to be executed simultaneously through a matrix of programmable logic blocks and interconnects. In this way, FPGAs enable asynchronous data processing. In addition, this matrix of programmable logic blocks can be customized depending on the task to be performed, giving FPGAs a wide range of applications.[182] These unique features make FPGAs particularly interesting for applications that require a high degree of parallelisation or special circuit adaptations, including the implementation of SNNs. In particular, the ability of FPGAs to create customized circuits for neural networks increases the efficiency of AI applications. This adaptability allows hardware to be optimized for the specific requirements of SNNs, which is not easily possible with conventional architectures[183] that are built from hard-wired integrated circuits also know as ASICs.

Another architecture that enables a efficient implementation of SNNs is neuromorphic hardware. This type of hardware attempts to imitate the structure and function of bilogical neurons. Neuromorphic hardware can have an analogue, digital or mixed-mode chip design. With analogue hardware, spiking neurons can be implemented using physical phenomena and electrical circuit components. The advantage of this approach is that operations that would be costly to implement as an explicit mathematical operation can be achieved very efficiently through the natural dynamics of the system.[184] In addition, physical variables can have

[180] Wang, S. P.: Computer Architecture and Organization: Fundamentals and Architecture Security (2021), pp. 4-6.

[181] Marković, D. et al.: Physics for neuromorphic computing (2020).

[182] Monmasson, E.; Cirstea, M. N.: FPGA design methodology for industrial control systems—A review (2007).

[183] Guo, W. et al.: Toward the optimal design and FPGA implementation of spiking neural networks (2021).

[184] Neil, D.; Liu, S.-C.: Effective sensor fusion with event-based sensors and deep network architectures (2016).

almost infinite precision. Most analogue hardware implementations differ in the extent to which analogue elements are used. Many analogue circuits perform only the computation of the neuron with analogue elements, leaving the communication of the spike signals digital.[185] Digital hardware implementations store the values of the neuron in binary bits. Therefore the precision of the variables depends on the number of bits used to represent them. This precision also has a strong influence on the energy consumption of the basic operations and the memory requirements for the storage of the variables. A advantage of digital designs over analogue hardware is that the precision of the variables can be controlled and guaranteed. In addition, digital hardware can be produced using established chip design and manufacturing techniques.[186]

Neuromorphic hardware offers significant advantages in terms of SNN implementation. By integrating memory and processing into a single computational unit, information can be transferred seamlessly between neurons without the bottleneck of data transfer between different subsystems. This direct connection enables more efficient processing of neural networks by reducing communication latency and increasing energy efficiency. In addition, the massive parallelisation of neuromorphic hardware allows a large amount of information to be processed simultaneously, which is particularly well suited to the asynchronous, event-driven nature of SNNs. This parallelisation not only accelerates the training of neural networks, but also enables faster inference and more responsive adaptation to changing environments. Neuromorphic hardware enables an optimized implementation of SNNs by overcoming the weaknesses of traditional von Neumann architectures. The direct fusion of memory and computing power, as well as massive parallelisation, helps to fully exploit the capabilities of SNNs.[187]

However, the production of neuromorphic hardware is a major challenge. A widely used circuit technology for computer hardware are complementary metal oxide semiconductors (CMOS). This type of semiconductor uses a p-type and an n-type metal oxide semiconductor, resulting in a circuit that consumes much less power than previous transistor units. As a result, CMOS has been used as the basic logic unit in modern computers since the 1980s. CMOS transistors can be mass-produced cheaply on silicon wafers using a lithographic process.[188] But CMOS technology has its limitations when it comes to implementing neuromorphic hardware. As a spiking neuron and a synapse are typically several micrometers wide, the number of neurons is limited by the area of the chip, resulting in a relatively small number of physical neurons. This is problematic because deep SNNs consist of millions of neurons. In addition, biological neurons are connected by several thousand synapses on average, which is difficult to realize with CMOS technology. In addition, the brain's high degree of interconnectivity, with its tree-like dimensional structure, is complicated to imitate with current two-dimensional electronics.[189]

Despite these challenges, several companies and research groups have already developed neuromorphic hardware. With Loihi 2 and the Lava framework, Intel is providing access to powerful neuromorphic hardware for research and development. Loihi 2 can simulate over 1 million neurons and 120 million synapses.[190] Other companies and universities are also pursuing the development of neuromorphic

[185] Camuñas-Mesa, L. A. et al.: Neuromorphic spiking neural networks (2019).

[186] Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022).

[187] Davies, M. et al.: Advancing neuromorphic computing with loihi: A survey of results and outlook (2021).

[188] Weste, N. H.; Eshraghian, K.: Principles of CMOS VLSI design: a systems perspective (1985), pp. 3-6.

[189] Marković, D. et al.: Physics for neuromorphic computing (2020).

[190] Intel Corporation: Taking Neuromorphic Computing to the Next Level with Loihi 2 (2021).

hardware, such as IBM TrueNorth[191], Neurogrid[192] from Stanford University, BrainScaleS-2[193] from Heidelberg University and SpiNNaker[194] from the University of Manchester. Even if neuromorphic hardware has great potential, its availability is still limited.

3.7 Existing Approaches

This chapter discusses existing approaches and pioneering work in the field of neuromorphic perception using SNNs in conjunction with lidar sensors. Progress has already been made in this area, and exploring existing approaches will help to develop a deeper understanding of the challenges and opportunities of neuromorphic perception based on lidar data.

Zhou and Wang proposed an SNN that directly uses temporal pulses from a lidar as input for object recognition. To do this, they created a custom dataset that projects the ToF of the lidar data onto a front view on a two-dimensional 16 x 16 plane. The information from the lidar was taken directly from the photodetector, where all the laser beams from the FoV were combined into a single photon detector array (SPDA). The custom database contains 3000 training images and 600 test images for 30 different classes. Different lighting and weather conditions were simulated by varying the noise levels of the images. For the object classification they used a feed-forward network with fully connected layers consisting of non-leaky integrate-and-fire neurons with exponentially decaying synaptic current kernels.[195] The change in membrane potential of the neuron model is described by equation 3-22.

$$\frac{du^j(t)}{dt} = \sum_i w_{ji} \sum_j \kappa(t - t_i^r) \tag{3-22}$$

The temporal derivation of the membrane potential u_j corresponds to the synaptic weight w_{ji} and the synaptic current kernel κ. If a spike arrives the neuron the synaptic current jumps instantaneously and then decays exponentially with time. Neurons that have already sent out an action potential are rested and are only allowed to spike again when a new image is presented to the network as input. Due to this special behavior of the individual neurons, it is possible to train the network using gradient decent. Zhou and Wang's research has shown that ToF can be used directly as an input. In addition, their proposed method is characterized by low latency and high accuracy even with increasing noise levels. However, it should be noted that this is a very simplified and abstract application example. The network processes only 16 x 16 gray-scale images, which are significantly smaller compared to lidar point clouds. Also their architecture is based on fully connected layers and does not include convolution, which could improve the generation of features. Furthermore, it only predicts the class of an object, which is not sufficient for an automated driving application.[195]

Some of these challenges were addressed in a follow-up paper by Zhou et al. They presented an approach that enables SNN object detection for more complex datasets. They used a spiking CNN with temporal

[191] Merolla, P. A. et al.: A million spiking-neuron integrated circuit (2014).

[192] Benjamin, B. V. et al.: Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations (2014).

[193] Pehle, C. et al.: The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity (2022).

[194] Furber, S. B. et al.: The spinnaker project (2014).

[195] Zhou, S.; Wang, W.: Object Detection based on LIDAR Temporal Pulses using Spiking Neural Networks (2018)

encoding as a feature extractor and combined it with the YOLOv2 architecture. Since real lidar point clouds require a suitable representation of the data in order to interpret them with an artificial network, they first quantified the 3D data of the KITTI benchmark dataset into voxels. The voxels were filled with hard-coded features. This was done by calculating the ToF of the points. Voxels that contain no lidar points were assigned a ToF of zero. If a voxel contained more than one point, the ToF of a randomly selected lidar point was taken. This resulted in a three-dimensional tensor with the shape 768 x 1024 x 21. To avoid the computational complexity of a 3D convolution, a 2D convolution with a 3x3x21 filter was used to match the depth of the voxelized point cloud.[196]

As in the previous work of Zhou and Wang, non-leaky integrate-and-fire neurons were used, combined with the same temporal encoding scheme. In total they used 9 spiking convolutional layers, 1 traditional convolutional layer, 5 max-pooling layers, and 3 intermediate layers. Skip connections have also been implemented, which are essentially extra connections between certain nodes and layers, allowing one or more layers of the network to be skipped. These skip connections facilitate the training of deep networks and have significantly improved the accuracy of the model. The model is based on YOLOv2 with the backbone replaced by a spiking CNN. Between these two parts, a conventional convolutional layer was inserted, which acts as an interface between the spiking part and the non-spiking part of the network. The network was trained using stochastic gradient decent.[196]

Compared to Complex-YOLO with the same number of convolution layers, the network achieves higher accuracy but at a lower FPS. However, the model of Zhou et al. is based on hard-coded features and only runs the convolutional part of the model with spiking neurons. Furthermore, the paper does not describe in detail how the connecting convolutional layer between the spiking and non-spiking parts of the network works. Nor does it go into detail about the implementation of skip connections, which have a significant impact on the accuracy of their model.[196]

Wang et al. have proposed a model consisting exclusively of spiking neurons that can process the lidar data from the KITTI dataset. In order to translate the lidar data into action potentials, an array of temporal pulses was reconstructed, similar to the work of Zhou and Wang[197] in 2018. The data of the point clouds can be transformed into an extended front view with equations 3-23 and 3-24 depending on the resolution of the lidar.[198]

$$x_{\text{front}} = arctan2(-y, x) \cdot \frac{180}{\pi \cdot Res_h} \tag{3-23}$$

$$y_{\text{front}} = -arctan2(z, d) \cdot \frac{180}{\pi \cdot Res_v} \tag{3-24}$$

[196] Zhou, S. et al.: Deep SCNN-Based real-time object detection for self-driving vehicles using LiDAR temporal data (2020)

[197] Zhou, S.; Wang, W.: Object Detection based on LIDAR Temporal Pulses using Spiking Neural Networks (2018).

[198] Wang, W. et al.: Temporal pulses driven spiking neural network for time and power efficient object recognition (2021).

Where the coordinates x, y, and z describe the points in the original point cloud and d represents the distance to the lidar sensor. Res_h and Res_v are the horizontal and vertical angular resolution of the lidar indicated in degrees. The model consists of two convolutional layers and two fully connected layers, both of which also used non-leaky integrate-and-fire neurons as in the previous papers. The output layer has 8 channels, corresponding to the number of classes in the KITTI dataset. Compared to other detection networks such as VGG-16 or ResNet-50, the SCNN achieves a higher accuracy in class detection with a shorter response time and a lower energy consumption of the model during interference. Even though the KITTI dataset was used as the basis for the lidar data, the task of the model was limited to object classification. Furthermore, the training process of the model is hardly addressed. A model that is used in the field of automated driving should be able to determine the pose as well as the class of the objects, for example through 3D bounding boxes.[199]

Vico et al. proposed a recurrent SNN for object recognition based on lidar data from the KITTI dataset. Recurrent networks are a special type of network that can remember patterns and information previously presented to it over time. The model consists of leaky integrate-and-fire neurons with layer-wise recurrent connections and is trained using a special backpropagation algorithm that uses surrogate gradients. They evaluated the classification performance with different input encodings. The best results were achieved with radial basis functions (RBF) encoding. They also developed an object extraction pipeline that creates 2D images of equal size, each containing an object from the KITTI dataset. The values of the pixels are derived from the depth of the 3D points. This approach is also limited to object recognition and therefore only determines the class of an object. Furthermore, the two-dimensional data obtained by transforming the KITTI dataset is more comparable to classical image data or RGB-D data obtained by combining camera and lidar sensors.[200]

The approaches presented in this section show that object detection with spiking networks based on lidar data is being researched. However, there are still some challenges to overcome to enable 3D object detection with models based purely on spiking neurons.

[199] Wang, W. et al.: Temporal pulses driven spiking neural network for time and power efficient object recognition (2021).

[200] Vicol, A.-D. et al.: Real-time classification of LIDAR data using discrete-time Recurrent Spiking Neural Networks (2022).

4 Methodology

This chapter presents the methodology of the master thesis, which aims to identify the potential and feasibility of a neuromorphic perception pipeline for 3D object detection based on lidar point clouds. For this purpose, a morphological analysis is chosen, which is a systematic approach to creative problem solving that includes a morphological box to comprehensively explore potential solutions. To apply this method, the perception pipeline is first decomposed into subcomponents. Different requirements are then defined for each component. These range from essential requirements, through secondary requirements, to recommended requirements that should enable the perception pipeline to exploit the full potential of neuromorphic technology. All subcomponents that meet the essential requirements of the neuromorphic perception pipeline are then entered into the morphological box. By combining the different components of the morphological box, concepts for a neuromorphic perception system are obtained that can fulfill the task of 3D object detection based on lidar data.

4.1 Definition of Requirements

Before defining the requirements for the individual components of the neuromorphic perception pipeline, we need to specify the intended task of the perception system. The perception pipeline should be able to perform 3D object detection based on lidar data, including estimation of class, pose, and dimension of objects in a given FoV. The pose and dimension of objects can be displayed using 3D bounding boxes, which would improve the comparability with other object detectors that usually generate bounding boxes as well. In addition to secondary requirements and recommended requirements, which are defined in the following, the essential requirements must be in line with the intended task. Accordingly, a concept of a neuromorphic perception pipeline that fulfills the essential requirements should be able to perform 3D object detection based point cloud data.

In order to define detailed requirements for each component of the perception pipeline, it is helpful to divide it into subsystems and subfunctions. Figure 4-1 shows a diagram of a neuromorphic perception pipeline. For each component of the system, we will identify the essential requirements that must be met in order to perform 3D object detection with SNNs based on lidar data, secondary requirements that do not have a direct impact on the fulfillment of the task but are desirable for economic or technical reasons, and recommended requirements that are more demanding in terms of implementation but necessary to exploit the full potential of the neuromorphic approach.

Starting with the lidar sensor, which must be able to generate a point cloud of the environment with sufficient range for use in automated driving. In addition to this essential requirement, a large field of view, preferably a 360 degree coverage of the environment, high resolution and detection range, and a low price are secondary requirements that are desirable. As we aim for a fast and efficient object detection system, it is recommended that the lidar sensor is able to output a high number of FPS. In addition, ToF lidars with direct access to the photodetector would accelerate the propagation of information, as the ToF could be encoded directly into spikes as the reflected laser beams return to the sensor. The next step in the perception pipeline is the encoding of the sensor data. The coding scheme used must convert the floating point values of the lidar into discrete spikes. To increase the energy efficiency of the neuromorphic perception system, a sparse and direct encoding of the ToF is recommended. We should exploit the time dimension of the spatio-temporal nature of the spikes, as this dimension is accompanied by a suitable

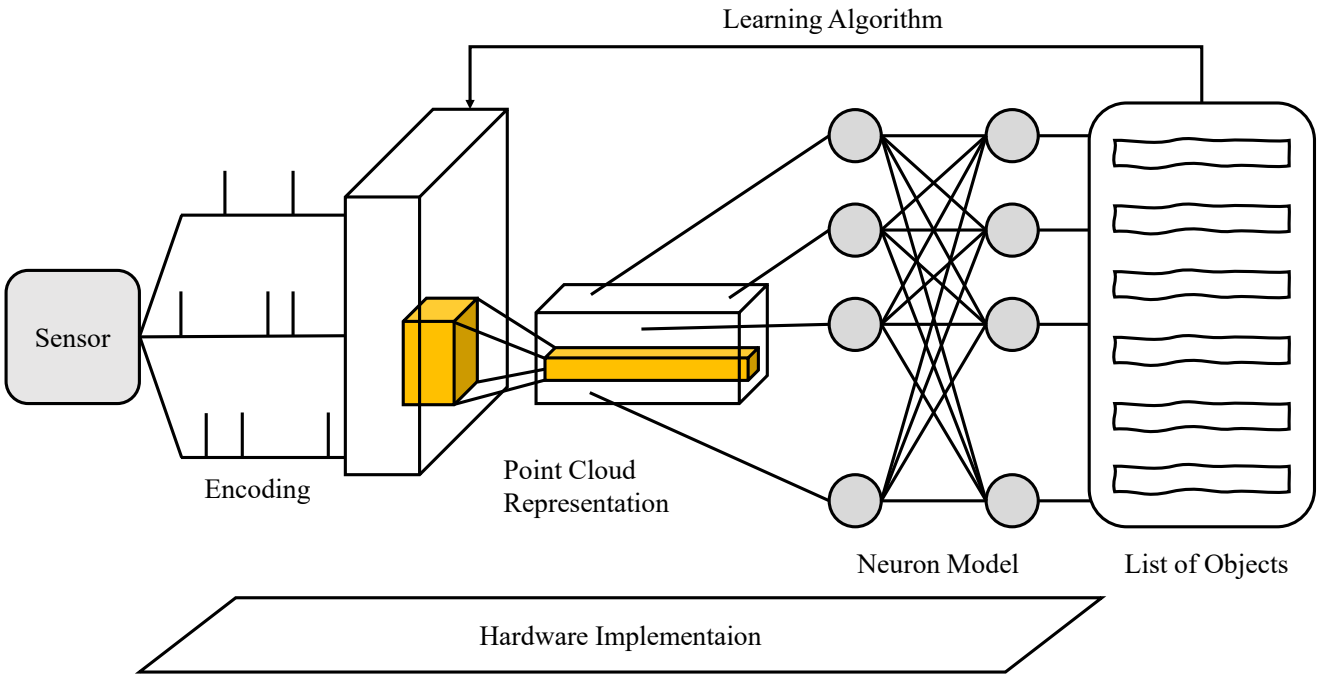

Figure 4-1: Concept of the neuromorphic perception pipeline.

hardware implementation without increased computational effort and thus energy consumption. The next step in the neuromorphic perception pipeline is the representation of the point cloud data. In order to perform 3D object detection with neural networks, we need to transform the unordered point set into a representation with uniform shape. In addition to this essential requirement, the possibility to use 2D object detectors would be an added value, as these frameworks are well known and well researched in computer vision. Furthermore, it is recommended that the generated features are not hard-coded, but that the feature detector can learn local and global features itself based on the dataset.

As a basic building block, we need a spiking neuron model that can mimic the fundamental behavior of biological neurons and is able propagate information using spikes. A neuron model with low computational complexity would be advantageous in order to limit the computational effort of the network and to accelerate the training. To investigate the effects of different neural dynamics on the detection network, a more complex neural model that can represent different types of biological neurons is recommended. This model should also be able to respond to a wide range of encoding methods. However, the real-time capability of the system must not be affected. A recommended spiking neuron model provides a compromise between these aspects. To accomplish the task of 3D object detection, we need to train a SNN with an appropriate learning algorithm. This algorithm must allow the training of a multi-layer SNN to match the complexity of the detection task. The adaptation of well established algorithms used to train ANNs could be a way to reduce the training effort and at the same time enables the use of already optimized training algorithms. Since we want to preserve the temporal domain of the SNN approach, learning algorithms that take into account the time-dependent behavior of SNNs are recommended. Finally, we need to specify requirements for the hardware on which the SNN will be implemented. The hardware must be able to compute the model equations and the training algorithms. From an economic point of view, hardware with a low price and good availability would be desirable. To take full advantage of the neuromorphic approach, it is

recommended to implement SNNs on dedicated hardware that allows asynchronous computation and provides direct interconnection between the computation and memory units. In addition, an analogue implementation of the neuron model used would be advantageous to further reduce the energy consumption of the perception system and improve the real-time capability of the pipeline.

4.2 Morphological Analysis

Having defined the task of the neuromorphic perception system and broken down the pipeline into different components, we now need a method to help us find partial solutions that fulfill the previously defined requirements. Since a neuromorphic perception pipeline is a complex system that is easier to solve if it is divided into smaller problems rather than viewed as a whole, we use a morphological analysis. Morphological analysis is a creative and heuristic approach to understanding complex problems. It allows us to consider all possible solutions in an unbiased way and to find new solutions by combining partial solutions. The main tool of this method is the morphological box, which is a matrix in which different features of a system or product are listed. By systematically combining and varying these features, new solutions and ideas can be generated, making the morphological box a useful tool for product development and innovation systems. Morphological analysis provides a systematic approach to solving scientific and engineering challenges and helps to identify innovation concepts for disruptive technologies.[201]

In figure 4-2 we see the morphological box that emerges from the subdivision of the perception system. The features of the morphological box follows the structure of the literature review in chapter 3. However,

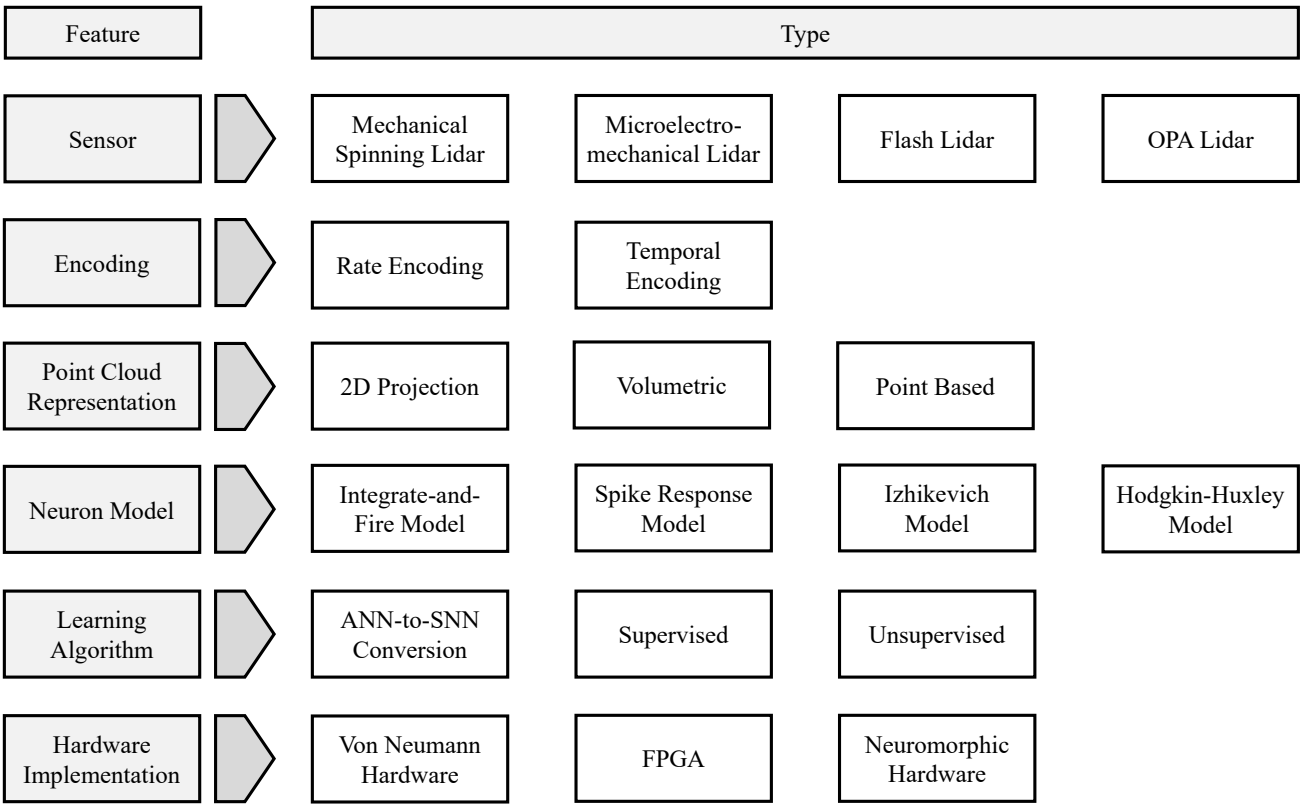

Figure 4-2: Morphological box with features and partial solutions of the neuromorphic perception pipeline.

[201] Zwicky, F.: Entdecken, erfinden, forschen im morphologischen Weltbild (1966).

this sequence of features can also be understood as the high-level structure of the neuromorphic perception pipeline. All types of the different features that meet the essential requirements are taken into account for the morphological box. The first feature of the morphological box is the sensor unit. In chapter 3.1 we discussed different types of lidars including, mechanical spinning lidars, MEMS lidars, flash lidars and OPA lidars. All these sensors have different properties but are able to capture a point cloud of the environment with sufficient range for automated driving, thus meeting the minimum requirements to be included in the morphological box. In order to process unstructured lidar data with neural networks, we need a suitable representation of the point cloud data. The next feature of the morphological box therefore includes the three approaches of point cloud representation consisting of 2D projection, volumetric and point-based representation. All three approaches allow us to transform the lidar data into a uniform shape to fulfill the task of 3D object detection with neural networks. Following the point cloud representation we have to choose a spiking neuron model. In chapter 3.4 we discussed several spiking neuron models, which differ mainly in their biological plausibility and computational complexity. As a essential requirement for the neuron model, we defined that it should exhibit the basic behavior of biological neurons and be able to propagate information through spikes. All four spiking neuron models in the morphological box, including the integrate-and-fire model, the spike response model, the Izhikevich model and the Hodgkin-Huxley model, meet this requirement. The next feature contains the three methods for training a SNN. For this purpose, we discussed ANN-to-SNN conversion, supervised and unsupervised learning algorithms in chapter 3.5. All three learning approaches are capable of training an SNN and thus fulfill the essential requirement for this feature. The final feature of the morphological box is the hardware implementation for SNNs. For this purpose, we considered various hardware architectures including von Neumann hardware representing CPUs and GPUs, FPGAs and neuromorphic hardware in chapter 3.6. All hardware designs are able to compute the model equations and the training algorithms, which is the essential requirement to run a neuromorphic system. In the following chapter, these features are combined to generate two possible solution paths.

The morphological analysis allows a systematic approach to creative problem solving, especially in the early stages of a system development. However, this method also has drawbacks. The results of morphological analysis are highly dependent on the choice of features and the subdivision of the system. Important aspects may be overlooked, which can lead to a limited solution space. On the other hand, too many features can be included in the morphological box, resulting in overly complex combinations and making the solution more difficult to comprehend. Another limitation of morphological analysis is the lack of direct visibility of the dependencies between different features. This can lead to unfeasible solutions when combining features that are not compatible with each other. As a result, prior knowledge of the subject and its interrelationships are necessary for a effective morphological analysis. Additionally, the parameters and features chosen for the morphological box are frequently subjective and susceptible to individual biases and opinions. Moreover, it should be acknowledged that morphological analysis only yields qualitative outcomes. A quantitative assessment of the solution path is not possible.

In order to minimize the weaknesses of the morphological box, this method can be combined with other creative problem solving techniques and quantitative evaluation methods to enable a more comprehensive and balanced solution finding. Another qualitative method is TRIZ, which is based on the study of inventive solutions described in patents. Like the morphological analysis it offers a systematic approach

to solving technical problems and can help to generate new solutions.[202] The aim of TRIZ is to resolve systemic contradictions. Instead of combining existing methods to arrive at a solution, this approach uses inventive techniques to address these contradictions. The first step is to break down the problem into generic subproblems that can be solved with generally applicable inventive methods. TRIZ should be used primarily when a system development problem cannot be solved with current methods. It is advisable to carry out a morphological analysis first and to use TRIZ only when unexpected problems arise during concept development.

As the system development progresses, a quantitative assessment of different solutions can be helpful in deciding between two solution paths. For this purpose, the iterative process of prototyping can be employed. It involves the creation of an initial model or prototype of the intended system. This can include the development of subfunctions that serve as the basis for the overall system. Prototyping is used to develop an early understanding of the system, which helps to identify problems before the actual development of the final system begins. This agile approach speeds up the development process and increases the likelihood of a successful end result. Prototyping can be performed based on a morphological analysis in order to test the feasibility of subfunctions.[203]

Due to the limited time frame and the complexity of the topic, this thesis will be limited to the application of a morphological analysis. The resulting concepts are to be understood as a starting point and can be investigated in more detail with the previous mentioned methods in order to further develop the concept of a neuromorphic perception pipeline based on lidar data.

[202] Savransky, S. D.: Engineering of creativity: Introduction to TRIZ methodology (2000), pp. 21-22.

[203] Budde, R. et al.: What is prototyping? (1990).

5 Results and Discussion

In this chapter, the results of the morphological analysis are described and discussed in detail. Two concepts are developed, each of which fulfills different requirements for a neuromorphic perception pipeline. We start with the baseline concept, which is based on the essential requirements. This concept should maximize the feasibility of a neuromorphic pipeline and combine as many secondary requirements as possible. The second concept is an advanced concept that meets the recommended requirements. This concept focuses on the energy saving potential that can be realized through the neuromorphic approach.

5.1 Baseline Concept

Starting with a baseline concept of a neuromorphic perception pipeline which is depicted in figure 5-1. This concept is primarily intended to demonstrate the feasibility of neuromorphic perception. Solutions and components that are already widely used or well researched will be preferred over those that further improve energy efficiency or better support the dynamics of spiking neurons. Based on the morphological box presented in the previous chapter we discuss step by step the illustrated solution path that matches the essential requirements needed to fulfill the task of 3D object detection based on lidar data. In addition, this concept attempts to meet as many secondary requirements of a neuromorphic perception system as possible, as far as the technical dependencies between the individual features of the morphological box allow.

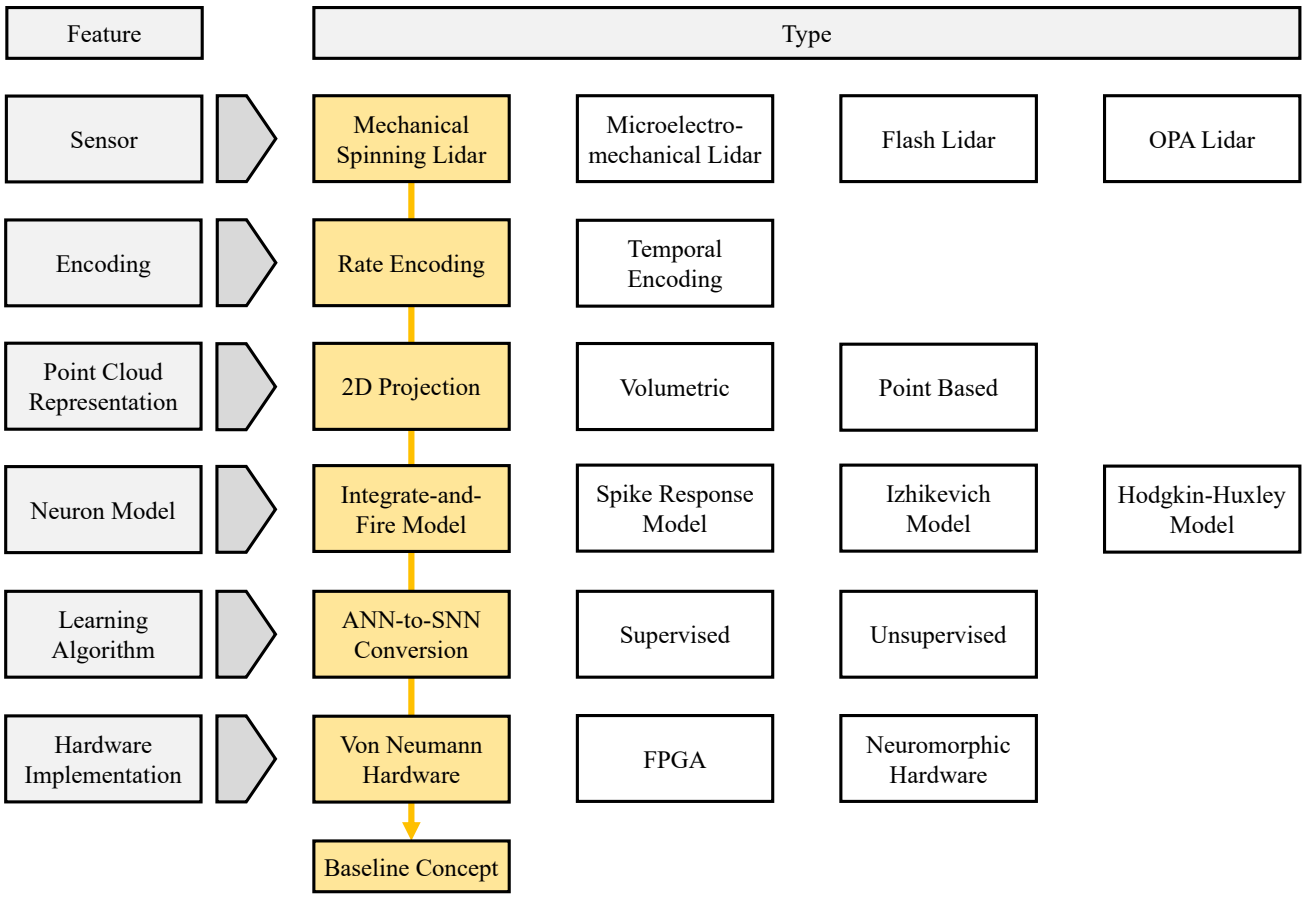

Figure 5-1: Baseline concept of a neuromorphic perception pipeline.

The first step in the morphological box is the selection of a sensor to obtain the necessary data of the environment. In this thesis we focus on the use of lidar sensors, as they provide rich spatial information about the environment and accurate distance measurements. This additional depth information enhances the overall robustness and reliability of the perception system. To fulfill the basic requirement we need to select a lidar sensor that can generate a point cloud of the environment with sufficient range for use in automated driving. All lidars in the morphological box are able to meet this requirement. In addition to the detection range a large FoV and high resolution are desirable. Therefore a mechanical spinning lidar was selected as this lidar design typically covers a FoV of 360 degrees. Even though this type of lidar is widely used in the automotive sector, it has certain disadvantages. Because the laser beams are distributed by a mechanical laser steering system, this sensor design is more sensitive to vibration than solid-state solutions. They also have a relatively low number of FPS, mainly due to the fact that most lidars only output data once the environment has been completely scanned. In addition, vehicle integration is challenging because these sensors take up more space than other designs, and to get a 360 degree view of the environment, the lidar should be positioned so that no other components or parts of the vehicle obstruct the sensor's view. In addition, the design of commercial vehicles is a key selling point, so car manufacturers try to install the sensors in a way that they are not directly visible. Despite these drawbacks, mechanical rotating lidars are commonly used in benchmark datasets, which underscores their ability to demonstrate the feasibility of a neuromorphic perceptual system.

The next step in the morphological box is the encoding of the lidar data into discrete spikes. It is crucial to take this process into consideration alongside the selected training algorithm. Since the baseline concept should be minimize the training effort, it is reasonable to use ANN detection frameworks and convert these networks to SNNs instead of developing networks from scratch based on spiking neurons. However, the ANN-to-SNN conversion relies on the use of rate coding schemes to approximate the continuous activation functions present in the underlying ANN. Furthermore, it is necessary to approximate the ToF of the lidar through a firing rate, resulting in a trade-off between accuracy and real-time capability. By defining a wider rate time window, a more accurate approximation of the ToF can be achieved. However, this results in a longer signal propagation time. The width of the time window for a certain discretisation accuracy depends on the clock rate of the hardware and the selected simulation step size of the neuron model. Instead using the count rate of single neuron we can encode the ToF by using a population rate of multiple neurons. This should theoretically allow us to narrow the time window without limiting accuracy.

After performing encoding, the representation approach for processing the unordered three-dimensional lidar data with machine learning methods must be selected. To maximize the feasibility of the baseline concept, it is reasonable to transform the point cloud into an image using a 2D projection, since 2D object detection frameworks are desired. It is important to acknowledge that converting a point cloud into a two-dimensional image results in some loss of information due to data compression by one dimension. Additionally, predicting 3D bounding boxes based on a 2D projection, especially from an extended front view, presents a challenging task. Accordingly, many projection methods transform the point cloud from a bird's eye perspective as this view offers improved recognition of object poses. Another option would be a volumetric approach using a filter in the first layer of the CNN that is as deep as the discretised point cloud is high. This would also allow the use of 2D detection frameworks, but more weights would need to be trained due to the larger filter in the first layer of the CNN compared to a 2D projection. 3D object detection networks based on 2D convolution layers can process data more quickly than frameworks relying

on computationally expensive 3D convolutions. This could potentially reduce the time delay caused by the calculation of the average spike rate introduced by the rate encoding.

Following the point cloud representation the selection of the spiking neuron model is the next step in the morphological box. For the baseline concept we select a integrate-and-fire model as this model is able to exhibit the fundamental neural dynamics of biological neurons and is characterized by low computing requirements. The basic leaky integrate-and-fire model requires only 5 FLOPS to simulate the neural dynamics over a single time step. More sophisticated versions of the integrate-and-fire model, such as the integrate-and-fire with adaptation and the quadratic or exponential integrate-and-fire model, require about 10 FLOPS per time step. Compared to the other neuron models listed in the morphological box, it is the least computationally intensive spiking neuron model. In addition, the integrate-and-fire model can be implement on analogue hardware as integration and leakage of the membrane potential can be modeled by subthreshold transistors and capacitors. However, the analogue implementation of integrate-and-fire model requires specific hardware that maps the electric circuit of the model.

The final component of the morphological box is the hardware of the neuromorphic perception system on which the software of the pipeline is implemented. The hardware must be able to solve the equations of the integrate-and-fire model and of the ANN-to-SNN conversion. For the baseline concept, we choose computer hardware based on the von Neumann architecture, as this hardware is sufficient to solve the system equations. In particular, GPUs offer adequate parallelisation capabilities, and since the individual neurons mostly require simple arithmetic operations, these computing units are preferable to CPUs. In addition, the von Neumann hardware allows both ANN and SNN to be computed, making it particularly suitable when an ANN-to-SNN conversion is used for training.

5.2 Advanced Concept

In this section, we discuss an advanced concept, as shown in figure 5-2, of a neuromorphic perception pipeline that aims to fully exploit the potential of the neuromorphic approach. This concept is based on the recommended requirements, which are technically and implementation-wise more demanding, but offer greater energy savings and more biologically plausible information processing. In the advanced concept, the secondary requirements are of lesser importance, as they do not directly support the neuromorphic approach and do not impact the feasibility of 3D object detection. Economic factors and a simpler technical implementation are not the primary focus of this concept.

The first step in the morphological box is the lidar sensor. In order to achieve a fast and efficient perception system, a low scan rate of the lidar sensor would be a limiting factor. Therefore, it is advisable to choose a lidar with a high number of FPS. Flash lidars are fitting for this purpose as they tend to have a higher scan rate than other lidar designs. In addition, flash lidars necessitate very little installation space, which makes them ideal for onboard accommodation. One option is to mount the lidar behind the windscreen in the upper region. This specific position offers a better visibility of the driving surroundings for example compared with an installation closer to the radiator grille. The latter could have limited detection performance of the system as small items nearby could obscure large parts of the FoV. However, it should be noted that flash lidars have a limited range and cannot provide a 360 degree view of the environment. Instead, they are capable of only perceiving a point cloud within a certain angle of view. Furthermore, the range of flash lidars is restricted since only one laser beam illuminates the environment at any given time. Nonetheless, in cases the photodetector can be accessed directly, the simultaneous illumination of the environment may

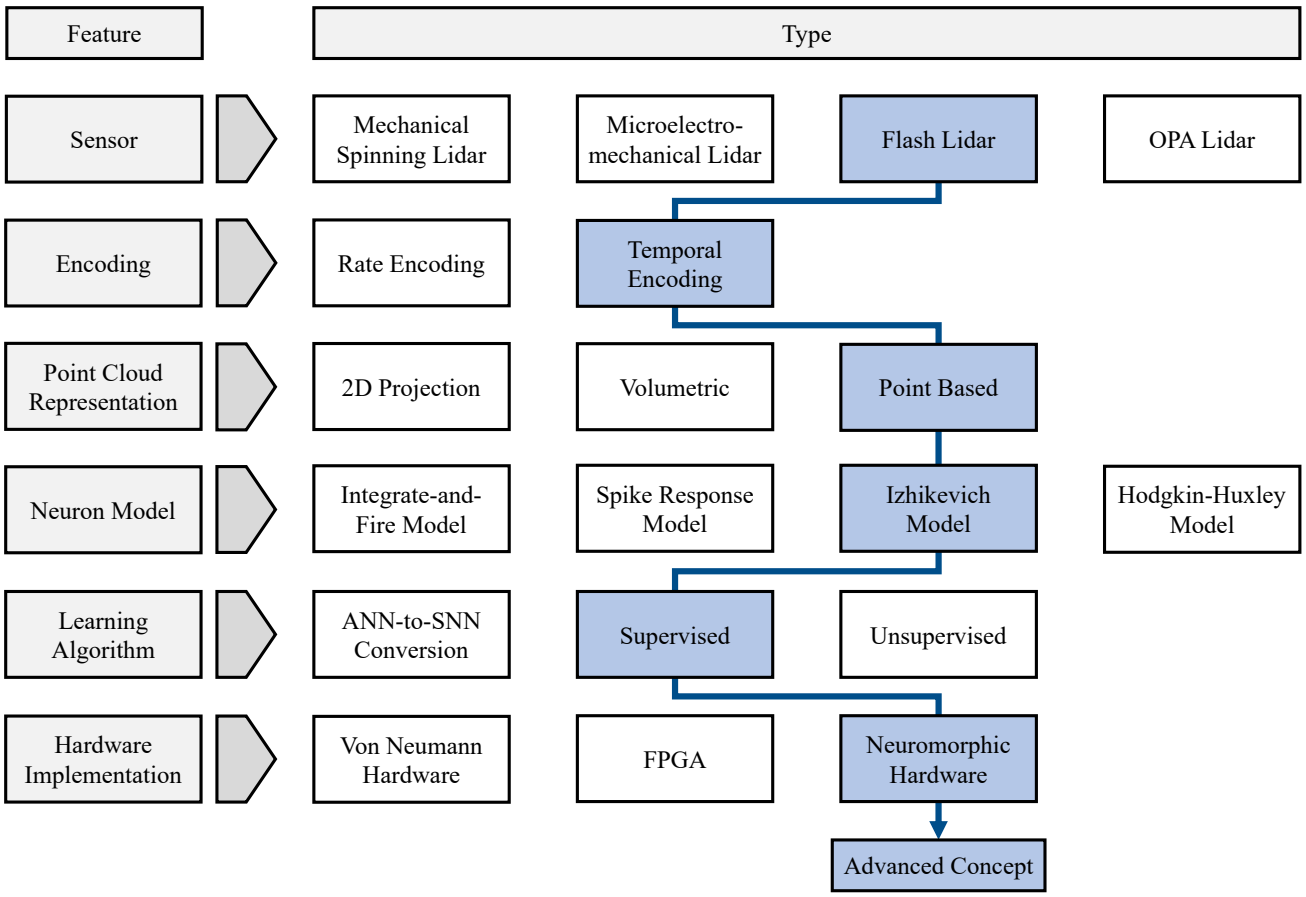

Figure 5-2: Advanced concept of a neuromorphic perception pipeline.

enable faster access to the ToF of the individual laser beams. As a single photon detector records each reflected laser beam, the sum of all laser beams can be promptly collected in form of an array containing the individual ToFs.

This leads us to the encoding of the lidar data, which is the next step in the morphological box. To increase the energy efficiency of the neuromorphic perception system, it is recommended to implement a sparse and direct encoding of the ToF using a temporal coding scheme. A TTFS coding scheme would be appropriate, where the global time reference is established when the laser beams are emitted. Temporal coding allows for the representation of ToF in a single spike, thereby consuming less energy as the energy demand of an SNN is proportional to the number of spikes that need to be propagated through the network. Objects that are distant and have a high ToF result in later spikes, and objects that are closer to the sensor and therefore have a lower ToF result in earlier spikes. By selecting a temporal coding scheme, there is no requirement to define a time window as in rate coding, which must be waited for before transmitting information. Rather, the ToF and thus the distance information corresponds directly with the elapsed time until the spikes occur.

The next feature in morphological box is the representation of the point cloud. We need to transform the unordered point cloud to further process the lidar data with machine learning methods. A representation of the lidar data is recommended that allows to learn local and global features from training data instead of predefining these features, e.g. in the form of a point cloud density. Therefore, representation types that extend the PointNet approach to non-segmented point clouds are a favored choice. For this purpose, we

discussed VoxelNet and PointPillar. The main difference between the two approaches is that PointPillars allows us to compute features using 2D convolution. Therefore, in contrast to the computationally intensive 3D convolution required to implement VoxelNet, it makes sense to use PointPillars as a representation of the lidar data. A challenge with both volumetric and point-based representation methods that discretise the point cloud into three-dimensional elements is that most of these elements do not contain lidar points, but are calculated using standard convolution. Especially the elements near the maximum range of the lidar sensor contain only a few points. This leads to a high computational demand when using ANNs which can be reduced by the use of sparse convolution. Further research is needed to determine the effect of empty voxels on convolution in SNNs, which is beyond the scope of this work. It is possible that sparse convolution is unnecessary because non-existent points within a voxel do not trigger spikes and therefore no value needs to be computed.

The point cloud representation is followed by the selection of a neuron model. To meet the requirements of the advanced concept, the spiking neuron model should be able to represent more complex neural dynamics and therefore have a higher biological plausibility, but still offer a moderate computational complexity that does not limit the real-time capability of the network. In section 3.4, we discussed the Hodgkin-Huxley model, which is able to simulate the spike generation based on sodium and potassium ion fluxes. This spiking neuron model offers the best biological plausibility of all the neuron models discussed in this thesis. However, the simulation of the electrophysiological properties of neurons comes at the cost of high computational complexity. The standard Hodgkin-Huxley model requires 1200 FLOPS for the simulation of one time step. Even the simplified version of the Hodgkin-Huxley model, using two instead of three differential equations, still requires several hundred FLOPS. This makes the Hodgkin-Huxley model unsuitable for simulating large networks. Instead of simulating the electrophysiological behavior of biological neurons like conductance-based neuron models such as the Hodgkin-Huxley model, we should also consider phenomenological neuron models such as the Izhikevich model. Phenomenological models describe when spikes occur and neglect the underlying principles. The shape of the spikes is approximated by a Dirac impulse, which is sufficient for the description of the neuron behavior. The phenomenological models also include the integrate-and-fire model, but the Izhikevich model is able to represent a wider range of neurons, which makes it a suitable model to study the influence of different neuronal dynamics on the performance of the detection network. Compared to the integrate-and-fire model, the Izhikevich model has only a slightly higher computational demand. To simulate a single time step, the model requires 13 FLOPS, which is only 3 FLOPS more than the integrate-and-fire model with additional adaptation and 8 FLOPS more than the basic leaky integrate-and-fire model. Due to its low computational requirements and excellent biological complexity, the Izhikevich model is highly suitable for the advanced concept of a neuromorphic perception pipeline.

The next step in the morphological box is the selection of a learning algorithm. Since we want to preserve the temporal domain of the SNN and the sparse coding of the lidar data using a temporal coding scheme, it is recommended to choose a learning algorithm that allows a direct learning of the SNN, which is not reliant on a rate coding scheme. One option would be to train the SNN in an unsupervised manner. Here, spike-timing-dependent plasticity (STDP), which is based on the Hebbian learning rule, could be an appropriate choice. STDP iteratively adjusts the synaptic weights of a neuron depending on the timing of the activity of neighboring neurons. This allows a SNN to form connections and recognize patterns in the data. Although STDP is biologically proven, it is difficult to train large SNNs because the

learning rule only considers the local information of neighboring neurons. To achieve convergence of large networks, we also need to consider the global information of the SNN architecture. A supervised learning method that uses both information domains is the spatio-temporal backpropagation (STBP) algorithm. This learning rule combines the time-dependent temporal domain and the layer-by-layer spatial domain to ensure the convergence of large SNNs, even when trained on complex datasets. Other supervised learning algorithms, such as SpikeProp or ST-SpikeProp, may not be as appropriate for training the advanced concept, due to their computationally expensive learning rules, leading to extended training durations, particularly for large-scale SNNs.

The last step in the morphological box is the selection of a suitable hardware implementation for the advanced neuromorphic perception pipeline. Our goal is to obtain neuromorphic object detector that consumes significantly less power than current ANN object detection frameworks. In order to leverage the energy efficiency potential of SNNs we strive for a highly parallel structure, sparse communication, and in-memory computation. These attributes are best fulfilled by neuromorphic hardware, as this type of hardware is optimized and specially developed for the execution of SNNs. Inspired by biological neurons in the human brain, neuromorphic hardware establishes massive parallelisation and asynchronous processing of the data, significantly reducing on-chip communication traffic and improving the energy efficiency of SNNs. On the other hand, hardware that is based on the Von Neumann architecture relies on sequential and central processing of the data, where processors and memory are separated resulting in high communication effort. While implementing SNNs is feasible, CPUs or GPUs present sub-optimal options. Although SNNs can be implemented on von Neumann hardware, the full potential of spiking communication cannot be exploited entirely by using CPUs or GPUs. Another optional hardware implementation are FPGAs, which offer great performance for many AI applications and support parallel and asynchronous processing, which especially useful for the implementation of SNNs. FPGAs were not developed directly for the implementation of SNNs, but offer a high degree of reconfigurability so that they can be used for various applications. However, neuromorphic hardware is specially designed for the calculation and execution of SNNs. For these reasons, the implementation of SNNs on neuromorphic hardware is highly recommended for the advanced concept of a neuromorphic perception system. The energy efficiency of neuromorphic hardware makes it an ideal candidate for embedded systems with power constraints, such as fully automated electric vehicles.

6 Conclusion and Outlook

This thesis has attempted to provide possible concepts for the realization of a neuromorphic perception system using the ToF of lidar data to overcome the massive energy consumption of current ANN perception systems. To achieve the necessary background knowledge, we discussed the basic principles of automated driving systems, artificial neural networks and spiking neural networks. Based on this knowledge, a comprehensive literature review was conducted, covering all the essential aspects required for the concept of a neuromorphic perception pipeline. First, we discussed different lidar technologies capable of providing point cloud data with sufficient range for automated driving. We then looked in detail at coding schemes that allow us to convert lidar data into discrete spikes that can be used as input to SNNs. The next chapter investigated different approaches to 3D object detection using point cloud data based on ANNs, starting with benchmark datasets needed for training the artificial networks and evaluating the detection performance, followed by possible representations of the point cloud that allow the processing of lidar data with machine learning methods. Next, an overview of spiking neuron models, which are able to mimic the dynamics of biological neurons, and different learning strategies suitable for SNNs were presented. Finally, the various hardware types capable of simulating SNNs and existing approaches using lidar data for neuromorphic object detection were discussed.

The results of the literature review were then evaluated using a morphological analysis. For this purpose, the neuromorphic perception system was subdivided into different components, including lidar sensors, data encoding, point cloud representations, spiking neuron models, learning algorithms and hardware implementations. For each feature various types that are discussed in the literature review where listed in a morphological box. By combining the different partial solutions in the morphological box, it was possible to develop two concepts of a neuromorphic perception pipeline that meet the previously defined requirements. The first concept tries to adapt existing methods and ideas from the field of 2D object detection with ANNs and extend them with the spatio-temporal domain of spiking neurons. The aim is to maximize the potential feasibility of neuromorphic 3D object detection. The baseline concept consists of a mechanical spinning lidar whose data is converted into spikes using a rate code. This is followed by a 2D projection of the point cloud, allowing the use of 2D detection frameworks that do not rely on computationally intensive 3D convolutions. To further limit the computational complexity of the baseline concept, the integrate-and-fire model is chosen as the spiking neuron model. The baseline object detector is trained using an ANN-to-SNN conversion that can be computed on conventional von Neumann hardware. However, this approach neglects important aspects of SNNs such as energy efficiency and low latency of data transmission. Therefore, a second concept has been investigated to address these aspects and fully exploit the potential of neuromorphic technology. This advanced concept uses a flash lidar with direct access to the photodetector to convert the ToF into spikes as quickly as possible using temporal encoding. To enable the SNN to learn its own features from the data, a point-based approach is chosen for the representation of the point cloud. As the basic building block of the SNN, we choose the Izhikevich model to investigate different neural dynamics. The network is trained directly with STBP on neuromorphic hardware, which significantly improves energy efficiency and computation time compared to other hardware solutions.

The aim of this thesis was to investigate the potential and feasibility of neuromorphic perception using ToF-based encoding of lidar data. Based on an extensive literature review and morphological analysis,

two concepts of neuromorphic perception were developed that demonstrate the great potential of this technology for automated driving. Although the two concepts presented are based on a comprehensive literature review, the feasibility of neuromorphic perception cannot be proven by morphological analysis alone. Further research is needed to demonstrate the feasibility of a neuromorphic perception system and to achieve the goal of a energy efficient and human-like automated driving system. However, when considering the state-of-the-art, it is evident that no scientific work has yet been published that implements neuromorphic perception based on lidar data with the level of complexity presented as in the advanced concept. The results of the thesis lay the groundwork for further research and development of neuromorphic perception based on lidar data.

Some aspects could not be discussed within the scope of this work. For example, the procedure for a point-based point cloud representation could not be investigated in detail. It should be researched how it is possible to use the point cloud of the lidar sensor directly as input of a SNN and to learn global and local features. An important aspect here is a appropriate pooling strategy, as only this can achieve the necessary symmetry of the point cloud, which enables the learning of unique features. Pooling is also important for the dimensional reduction of data in the backbone, whereby the most significant data should be retained. In the case of spiking networks, pooling is therefore heavily dependent on the data encoding used. One possibility is to develop a SNN prototype that is able to consume segmented point clouds representing only one object in order to simplify the challenge. Later, this object detector can be extended to point clouds that are relevant for automated driving and represent an entire driving situation. In particular, the use of sparse convolution layers should be investigated, as this can significantly accelerate the convolution. At an advanced stage of development, the influence of different neural dynamics and how they affect the performance of object detection can also be investigated. The Izhikevich model would be particularly useful here, as it is able to simulate different neurons with a manageable computational effort. It would also be interesting to investigate the influence of the hardware on energy consumption and inference time of the perception pipeline. In particular, the implementation of SNNs on FPGAs is a promising alternative to neuromorphic hardware. This is certainly only a selection of topics that are suitable for further research, but the points mentioned can be a starting point for exploring neuromorphic perception for automated driving.

This thesis demonstrated the enormous potential of neuromorphic perception for automated driving. It is essential to drastically reduce the energy consumption of automated driving systems in order to realize the vision of automated mobility. In particular, the shift to electric mobility and the transition to a sustainable energy supply will further limit the energy consumption of an automated driving system. Electrical energy is a precious commodity and should be used wisely. Therefore, it is strongly recommended to further research the implementation of neuromorphic perception to make road transport more efficient, sustainable and above all safer in the future.

Bibliography

Aamir, S. A. et al.: An accelerated LIF neuronal network array (2018)
Aamir, Syed Ahmed; Stradmann, Yannik; Müller, Paul; Pehle, Christian; Hartel, Andreas; Grübl, Andreas; Schemmel, Johannes,; Meier, Karlheinz: An accelerated LIF neuronal network array for a large-scale mixed-signal neuromorphic architecture, in: IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 65, pp. 4299–4312, 2018

Adrian, E. D. et al.: The impulses produced by sensory nerve-endings (1926)
Adrian, Edgar D; Zotterman, Yngve: The impulses produced by sensory nerve-endings: Part II. The response of a Single End-Organ, in: The Journal of physiology, Vol. 61, p. 151, 1926

Ahmed, H. U. et al.: Technology developments and impacts of connected and autonomous vehicles: An overview (2022)
Ahmed, Hafiz Usman; Huang, Ying; Lu, Pan,; Bridgelall, Raj: Technology developments and impacts of connected and autonomous vehicles: An overview, in: Smart Cities, Vol. 5, pp. 382–404, 2022

Akai, N. et al.: Robust localization using 3D NDT scan matching (2017)
Akai, Naoki; Morales, Luis Yoichi; Takeuchi, Eijiro; Yoshihara, Yuki,; Ninomiya, Yoshiki: Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching, in: 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, pp. 1356–1363, 2017

Albawi, S. et al.: Understanding of a convolutional neural network (2017)
Albawi, Saad; Mohammed, Tareq Abed,; Al-Zawi, Saad: Understanding of a convolutional neural network, in: 2017 international conference on engineering and technology (ICET), Ieee, pp. 1–6, 2017

Amzajerdian, F. et al.: Imaging flash lidar for autonomous safe landing and spacecraft proximity operation (2016)
Amzajerdian, Farzin; Roback, Vincent E; Bulyshev, Alexander; Brewster, Paul F,; Hines, Glenn D: Imaging flash lidar for autonomous safe landing and spacecraft proximity operation, in: (Hrsg.): AIAA SPACE 2016, 2016

Anderson, J. A.: A simple neural network generating an interactive memory (1972)
Anderson, James A: A simple neural network generating an interactive memory, in: Mathematical biosciences, Vol. 14, pp. 197–220, 1972

Arnold, E. et al.: A survey on 3d object detection methods for autonomous driving applications (2019)
Arnold, Eduardo; Al-Jarrah, Omar Y; Dianati, Mehrdad; Fallah, Saber; Oxtoby, David,; Mouzakitis, Alex: A survey on 3d object detection methods for autonomous driving applications, in: IEEE Transactions on Intelligent Transportation Systems, Vol. 20, pp. 3782–3795, 2019

Auge, D. et al.: A survey of encoding techniques for signal processing in spiking neural networks (2021)
Auge, Daniel; Hille, Julian; Mueller, Etienne,; Knoll, Alois: A survey of encoding techniques for signal processing in spiking neural networks, in: Neural Processing Letters, Vol. 53, pp. 4693–4710, 2021

Bacha, A. et al.: Odin: Team victortango's entry in the darpa urban challenge (2008)
Bacha, Andrew; Bauman, Cheryl; Faruque, Ruel; Fleming, Michael; Terwelp, Chris; Reinholtz, Charles; Hong, Dennis; Wicks, Al; Alberi, Thomas; Anderson, David, et al.: Odin: Team victortango's entry in the darpa urban challenge, in: Journal of field Robotics, Vol. 25, pp. 467–492, 2008

Badue, C. et al.: Self-driving cars: A survey (2021)
Badue, Claudine; Guidolini, Rânik; Carneiro, Raphael Vivacqua; Azevedo, Pedro; Cardoso, Vinicius B.; Forechi, Avelino; Jesus, Luan; Berriel, Rodrigo; Paixão, Thiago M.; Mutz, Filipe; Paula Veronese, Lucas de; Oliveira-Santos, Thiago,; Souza, Alberto F. De: Self-driving cars: A survey, in: Expert Systems with Applications, Vol. 165, 2021

Behroozpour, B. et al.: Lidar system architectures and circuits (2017)
Behroozpour, Behnam; Sandborn, Phillip AM; Wu, Ming C,; Boser, Bernhard E: Lidar system architectures and circuits, in: IEEE Communications Magazine, Vol. 55, pp. 135–142, 2017

Beltrán, J. et al.: Birdnet: a 3d object detection framework from lidar information (2018)
Beltrán, Jorge; Guindel, Carlos; Moreno, Francisco Miguel; Cruzado, Daniel; Garcia, Fernando,; De La Escalera, Arturo: Birdnet: a 3d object detection framework from lidar information, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE, pp. 3517–3523, 2018

Benjamin, B. V. et al.: Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations (2014)
Benjamin, Ben Varkey; Gao, Peiran; McQuinn, Emmett; Choudhary, Swadesh; Chandrasekaran, Anand R; Bussat, Jean-Marie; Alvarez-Icaza, Rodrigo; Arthur, John V; Merolla, Paul A,; Boahen, Kwabena: Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, in: Proceedings of the IEEE, Vol. 102, pp. 699–716, 2014

Bierstedt, J. et al.: Effects of next-generation vehicles on travel demand and highway capacity (2014)
Bierstedt, Jane; Gooze, Aaron; Gray, Chris; Peterman, Josh; Raykin, Leon,; Walters, Jerry: Effects of next-generation vehicles on travel demand and highway capacity, in: FP Think Working Group, Vol. 8, pp. 10–11, 2014

Bohte, S. M. et al.: Error-backpropagation in temporally encoded networks of spiking neurons (2002)
Bohte, Sander M; Kok, Joost N,; La Poutre, Han: Error-backpropagation in temporally encoded networks of spiking neurons, in: Neurocomputing, Vol. 48, pp. 17–37, 2002

Budde, R. et al.: What is prototyping? (1990)
Budde, Reinhard; Kautz, Karlheinz; Kuhlenkamp, Karin,; Züllighoven, Heinz: What is prototyping?, in: Information Technology & People, Vol. 6, pp. 89–95, 1990

Caesar, H. et al.: nuscenes: A multimodal dataset for autonomous driving (2020)
Caesar, Holger; Bankiti, Varun; Lang, Alex H; Vora, Sourabh; Liong, Venice Erin; Xu, Qiang; Krishnan, Anush; Pan, Yu; Baldan, Giancarlo,; Beijbom, Oscar: nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11621–11631, 2020

Camuñas-Mesa, L. A. et al.: Neuromorphic spiking neural networks (2019)
Camuñas-Mesa, Luis A; Linares-Barranco, Bernabé,; Serrano-Gotarredona, Teresa: Neuromorphic spiking neural networks and their memristor-CMOS hardware implementations, in: Materials, Vol. 12, p. 2745, 2019

Cheng, X. et al.: Learning depth with convolutional spatial propagation network (2019)
Cheng, Xinjing; Wang, Peng,; Yang, Ruigang: Learning depth with convolutional spatial propagation network, in: IEEE transactions on pattern analysis and machine intelligence, Vol. 42, pp. 2361–2379, 2019

Chi, L. et al.: Deep steering: Learning end-to-end driving model from spatial and temporal visual cues (2017)
Chi, Lu; Mu, Yadong: Deep steering: Learning end-to-end driving model from spatial and temporal visual cues, in: arXiv preprint arXiv:1708.03798, 2017

Citri, A. et al.: Synaptic plasticity: multiple forms, functions, and mechanisms (2008)
Citri, Ami; Malenka, Robert C: Synaptic plasticity: multiple forms, functions, and mechanisms, in: Neuropsychopharmacology, Vol. 33, pp. 18–41, 2008

Commission, I. E. et al.: Safety of laser products-Part 1: Equipment classification and requirements (2007)
Commission, International Electrotechnical et al.: Safety of laser products-Part 1: Equipment classification and requirements, in: IEC 60825-1, 2007

Connors, B. W. et al.: Intrinsic firing patterns of diverse neocortical neurons (1990)
Connors, Barry W; Gutnick, Michael J: Intrinsic firing patterns of diverse neocortical neurons, in: Trends in neurosciences, Vol. 13, pp. 99–104, 1990

Conrad, B.: Audi stoppt Pläne für teilautonomen A8 (2020)
Conrad, Bernd: Audi stoppt Pläne für teilautonomen A8, URL: https://www.auto-motor-und-sport.de/tech-zukunft/audi-a8-autonom-level-3-gesetz-grundlage/, 2020, visited on 07/07/2023

Creutzfeldt, O. D.: Generality of the functional structure of the neocortex (1977)
Creutzfeldt, Otto D: Generality of the functional structure of the neocortex, in: Naturwissenschaften, Vol. 64, pp. 507–517, 1977

Cruise LLC: Driving cities forward (2023)
Cruise LLC: Driving cities forward, URL: https://getcruise.com/about/, 2023, visited on 11/08/2023

Davies, M. et al.: Loihi: A neuromorphic manycore processor with on-chip learning (2018)
Davies, Mike; Srinivasa, Narayan; Lin, Tsung-Han; Chinya, Gautham; Cao, Yongqiang; Choday, Sri Harsha; Dimou, Georgios; Joshi, Prasad; Imam, Nabil; Jain, Shweta, et al.: Loihi: A neuromorphic manycore processor with on-chip learning, in: Ieee Micro, Vol. 38, pp. 82–99, 2018

Davies, M. et al.: Advancing neuromorphic computing with loihi: A survey of results and outlook (2021)
Davies, Mike; Wild, Andreas; Orchard, Garrick; Sandamirskaya, Yulia; Guerra, Gabriel A Fonseca; Joshi, Prasad; Plank, Philipp,; Risbud, Sumedh R: Advancing neuromorphic computing with loihi: A survey of results and outlook, in: Proceedings of the IEEE, Vol. 109, pp. 911–934, 2021

Deng, J. et al.: Imagenet: A large-scale hierarchical image database (2009)
Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai,; Fei-Fei, Li: Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, pp. 248–255, 2009

Diehl, P. U. et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing (2015)
Diehl, Peter U; Neil, Daniel; Binas, Jonathan; Cook, Matthew; Liu, Shih-Chii,; Pfeiffer, Michael: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: 2015 International joint conference on neural networks (IJCNN), ieee, pp. 1–8, 2015

Dosovitskiy, A. et al.: CARLA: An open urban driving simulator (2017)
Dosovitskiy, Alexey; Ros, German; Codevilla, Felipe; Lopez, Antonio,; Koltun, Vladlen: CARLA: An open urban driving simulator, in: Conference on robot learning, PMLR, pp. 1–16, 2017

Engelcke, M. et al.: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks (2017)
Engelcke, Martin; Rao, Dushyant; Wang, Dominic Zeng; Tong, Chi Hay,; Posner, Ingmar: Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 1355–1361, 2017

Fahlman, S. E. et al.: An empirical study of learning speed in back-propagation networks (1988)
Fahlman, Scott E et al.: An empirical study of learning speed in back-propagation networks, Carnegie Mellon University, Computer Science Department Pittsburgh, PA, USA, 1988

Feng, D. et al.: Towards safe autonomous driving (2018)
Feng, Di; Rosenbaum, Lars,; Dietmayer, Klaus: Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection, in: 2018 21st international conference on intelligent transportation systems (ITSC), IEEE, pp. 3266–3273, 2018

FitzHugh, R.: Impulses and physiological states in theoretical models of nerve membrane (1961)
FitzHugh, Richard: Impulses and physiological states in theoretical models of nerve membrane, in: Biophysical journal, Vol. 1, pp. 445–466, 1961

Furber, S. B. et al.: The spinnaker project (2014)
Furber, Steve B; Galluppi, Francesco; Temple, Steve,; Plana, Luis A: The spinnaker project, in: Proceedings of the IEEE, Vol. 102, pp. 652–665, 2014

Gaidon, A. et al.: Virtual worlds as proxy for multi-object tracking analysis (2016)
Gaidon, Adrien; Wang, Qiao; Cabon, Yohann,; Vig, Eleonora: Virtual worlds as proxy for multi-object tracking analysis, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4340–4349, 2016

Gautrais, J. et al.: Rate coding versus temporal order coding: a theoretical approach (1998)
Gautrais, Jacques; Thorpe, Simon: Rate coding versus temporal order coding: a theoretical approach, in: Biosystems, Vol. 48, pp. 57–65, 1998

Geiger, A. et al.: Are we ready for autonomous driving? the kitti vision benchmark suite (2012)
Geiger, Andreas; Lenz, Philip,; Urtasun, Raquel: Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE conference on computer vision and pattern recognition, IEEE, pp. 3354–3361, 2012

Gerstner, W.: Time structure of the activity in neural network models (1995)
Gerstner, Wulfram: Time structure of the activity in neural network models, in: Physical review E, Vol. 51, p. 738, 1995

Gerstner, W.; Kistler, W. M.: Spiking neuron models: Single neurons, populations, plasticity (2002)
Gerstner, Wulfram; Kistler, Werner M: Spiking neuron models: Single neurons, populations, plasticity, Cambridge university press, 2002

Glorot, X. et al.: Deep sparse rectifier neural networks (2011)
Glorot, Xavier; Bordes, Antoine,; Bengio, Yoshua: Deep sparse rectifier neural networks, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, pp. 315–323, 2011

Guo, W. et al.: Toward the optimal design and FPGA implementation of spiking neural networks (2021)
Guo, Wenzhe; Yantır, Hasan Erdem; Fouda, Mohammed E; Eltawil, Ahmed M,; Salama, Khaled Nabil: Toward the optimal design and FPGA implementation of spiking neural networks, in: IEEE Transactions on Neural Networks and Learning Systems, Vol. 33, pp. 3988–4002, 2021

Hamanaka, H. et al.: Quantized spiking neuron with A/D conversion functions (2006)
Hamanaka, Hiroshi; Torikai, Hiroyuki,; Saito, Toshimichi: Quantized spiking neuron with A/D conversion functions, in: IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 53, pp. 1049–1053, 2006

Hebb, D. O.: The organization of behavior: A neuropsychological theory (2005)
Hebb, Donald Olding: The organization of behavior: A neuropsychological theory, Psychology press, 2005

Hebermehl, G. et al.: Mercedes mit Level-3-Zulassung (2023)
Hebermehl, Gregor; Baumann, Uli: Mercedes mit Level-3-Zulassung, https://www.auto-motor-und-sport.de/tech-zukunft/mercedes-autonom-level-3-drive-pilot-haftung-unfall/, 2023

Hinton, G. E. et al.: Reducing the dimensionality of data with neural networks (2006)
Hinton, Geoffrey E; Salakhutdinov, Ruslan R: Reducing the dimensionality of data with neural networks, in: science, Vol. 313, pp. 504–507, 2006

Hodgkin, A. L. et al.: A quantitative description of membrane current (1952)
Hodgkin, Alan L; Huxley, Andrew F: A quantitative description of membrane current and its application to conduction and excitation in nerve, in: The Journal of physiology, Vol. 117, p. 500, 1952

Hopfield, J. J.: Pattern recognition computation using action potential timing for stimulus representation (1995)
Hopfield, John J: Pattern recognition computation using action potential timing for stimulus representation, in: Nature, Vol. 376, pp. 33–36, 1995

Hosang, J. et al.: Learning non-maximum suppression (2017)
Hosang, Jan; Benenson, Rodrigo,; Schiele, Bernt: Learning non-maximum suppression, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4507–4515, 2017

Intel Corporation: Taking Neuromorphic Computing to the Next Level with Loihi 2 (2021)
Intel Corporation: Taking Neuromorphic Computing to the Next Level with Loihi 2, in: Intel Technology Brief, 2021

Izhikevich, E. M.: Simple model of spiking neurons (2003)
Izhikevich, Eugene M: Simple model of spiking neurons, in: IEEE Transactions on neural networks, Vol. 14, pp. 1569–1572, 2003

Javanshir, A. et al.: Advancements in algorithms and neuromorphic hardware for spiking neural networks (2022)
Javanshir, Amirhossein; Nguyen, Thanh Thi; Mahmud, MA Parvez,; Kouzani, Abbas Z: Advancements in algorithms and neuromorphic hardware for spiking neural networks, in: Neural Computation, Vol. 34, pp. 1289–1328, 2022

Jocher, G. et al.: ultralytics/yolov5: v3. 0 (2020)
Jocher, Glenn; Stoken, Alex; Borovec, Jirka; Changyu, Liu; Hogan, Adam; Diaconu, Laurentiu; Poznanski, Jake; Yu, Lijun; Rai, Prashant; Ferriday, Russ, et al.: ultralytics/yolov5: v3. 0, in: Zenodo, 2020

Johansson, R. S. et al.: First spikes in ensembles of human tactile afferents (2004)
Johansson, Roland S; Birznieks, Ingvars: First spikes in ensembles of human tactile afferents code complex spatial fingertip events, in: Nature neuroscience, Vol. 7, pp. 170–177, 2004

Kasabov, N. K.: Time-space, spiking neural networks and brain-inspired artificial intelligence (2019)
Kasabov, Nikola K: Time-space, spiking neural networks and brain-inspired artificial intelligence, Springer, 2019

Keysers, C. et al.: The speed of sight (2001)
Keysers, Christian; Xiao, D-K; Földiák, Peter,; Perrett, David I: The speed of sight, in: Journal of cognitive neuroscience, Vol. 13, pp. 90–101, 2001

Krizhevsky, A. et al.: Imagenet classification with deep convolutional neural networks (2012)
Krizhevsky, Alex; Sutskever, Ilya,; Hinton, Geoffrey E: Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, Vol. 25, 2012

Lai, K. et al.: A large-scale hierarchical multi-view rgb-d object dataset (2011)
Lai, Kevin; Bo, Liefeng; Ren, Xiaofeng,; Fox, Dieter: A large-scale hierarchical multi-view rgb-d object dataset, in: 2011 IEEE international conference on robotics and automation, IEEE, pp. 1817–1824, 2011

Lambert, J. et al.: Tsukuba challenge 2017 dynamic object tracks dataset for pedestrian behavior analysis (2018)
Lambert, Jacob; Liang, Leslie; Morales, Luis Yoichi; Akai, Naoki; Carballo, Alexander; Takeuchi, Eijiro; Narksri, Patiphon; Seiya, Shunya,; Takeda, Kazuya: Tsukuba challenge 2017 dynamic object tracks dataset for pedestrian behavior analysis, in: Journal of Robotics and Mechatronics, Vol. 30, pp. 598–612, 2018

Lang, A. H. et al.: Pointpillars: Fast encoders for object detection from point clouds (2019)
Lang, Alex H; Vora, Sourabh; Caesar, Holger; Zhou, Lubing; Yang, Jiong,; Beijbom, Oscar: Pointpillars: Fast encoders for object detection from point clouds, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12697–12705, 2019

LeCun, Y. et al.: Deep learning (2015)
LeCun, Yann; Bengio, Yoshua,; Hinton, Geoffrey: Deep learning, in: Nature, Vol. 521, pp. 436–444, 2015

Lennie, P.: The cost of cortical computation (2003)
Lennie, Peter: The cost of cortical computation, in: Current biology, Vol. 13, pp. 493–497, 2003

Li, B.: 3d fully convolutional network for vehicle detection in point cloud (2017)
Li, Bo: 3d fully convolutional network for vehicle detection in point cloud, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 1513–1518, 2017

Li, B. et al.: Vehicle detection from 3d lidar using fully convolutional network (2016)
Li, Bo; Zhang, Tianlei,; Xia, Tian: Vehicle detection from 3d lidar using fully convolutional network, in: arXiv preprint arXiv:1608.07916, 2016

Li, Y. et al.: Lidar for Autonomous Driving: The Principles, Challenges, and Trends (2020)
Li, You; Ibanez-Guzman, Javier: Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems, in: IEEE Signal Processing Magazine, Vol. 37, pp. 50–61, 2020

Liang, W. et al.: A survey of 3D object detection (2021)
Liang, Wei; Xu, Pengfei; Guo, Ling; Bai, Heng; Zhou, Yang,; Chen, Feng: A survey of 3D object detection, in: Multimedia Tools and Applications, Vol. 80, pp. 29617–29641, 2021

Ma, X. et al.: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving (2019)
Ma, Xinzhu; Wang, Zhihui; Li, Haojie; Zhang, Pengbo; Ouyang, Wanli,; Fan, Xin: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6851–6860, 2019

Maass, W.: Networks of spiking neurons: The third generation of neural network models (1997)
Maass, Wolfgang: Networks of spiking neurons: The third generation of neural network models, Vol. 10, pp. 1659–1671, 1997

Mahesh, B.: Machine learning algorithms-a review (2020)
Mahesh, Batta: Machine learning algorithms-a review, in: International Journal of Science and Research (IJSR).[Internet], Vol. 9, pp. 381–386, 2020

Marković, D. et al.: Physics for neuromorphic computing (2020)
Marković, Danijela; Mizrahi, Alice; Querlioz, Damien,; Grollier, Julie: Physics for neuromorphic computing, in: Nature Reviews Physics, Vol. 2, pp. 499–510, 2020

Masquelier, T. et al.: Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains (2008)
Masquelier, Timothée; Guyonneau, Rudy,; Thorpe, Simon J: Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains, in: PloS one, Vol. 3, e1377, 2008

Masquelier, T. et al.: Unsupervised learning of visual features through spike timing dependent plasticity (2007)
Masquelier, Timothée; Thorpe, Simon J: Unsupervised learning of visual features through spike timing dependent plasticity, in: PLoS computational biology, Vol. 3, e31, 2007

McAllister, R. et al.: Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning (2017)
McAllister, RT; Gal, Yarin; Kendall, Alex; Van Der Wilk, Mark; Shah, Amar; Cipolla, Roberto,; Weller, Adrian: Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning, in: International Joint Conferences on Artificial Intelligence, Inc., 2017

McCulloch, W. S. et al.: A logical calculus of the ideas immanent in nervous activity (1943)
McCulloch, Warren S; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity, in: The bulletin of mathematical biophysics, Vol. 5, pp. 115–133, 1943

McManamon, P. F. et al.: Optical phased array technology (1996)
McManamon, Paul F; Dorschner, Terry A; Corkum, David L; Friedman, Larry J; Hobbs, Douglas S; Holz, Michael; Liberman, Sergey; Nguyen, Huy Q; Resler, Daniel P; Sharp, Richard C, et al.: Optical phased array technology, in: Proceedings of the IEEE, Vol. 84, pp. 268–298, 1996

Merolla, P. A. et al.: A million spiking-neuron integrated circuit (2014)
Merolla, Paul A; Arthur, John V; Alvarez-Icaza, Rodrigo; Cassidy, Andrew S; Sawada, Jun; Akopyan, Filipp; Jackson, Bryan L; Imam, Nabil; Guo, Chen; Nakamura, Yutaka, et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface, in: Science, Vol. 345, pp. 668–673, 2014

Mink, J. W. et al.: Ratio of central nervous system to body metabolism in vertebrates (1981)
Mink, Jonathan W; Blumenschine, Robert J,; Adams, David B: Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis, in: American Journal of Physiology-Regulatory, Integrative and Comparative Physiology, Vol. 241, R203–R212, 1981

Minsky, M.; Papert, S. A.: Perceptrons: an introduction to computational geometry (2017)
Minsky, Marvin; Papert, Seymour A: Perceptrons, reissue of the 1988 expanded edition with a new foreword by Léon Bottou: an introduction to computational geometry, MIT press, 2017

Monmasson, E. et al.: FPGA design methodology for industrial control systems—A review (2007)
Monmasson, Eric; Cirstea, Marcian N: FPGA design methodology for industrial control systems—A review, in: IEEE transactions on industrial electronics, Vol. 54, pp. 1824–1842, 2007

Montemerlo, M. et al.: Junior: The stanford entry in the urban challenge (2008)
Montemerlo, Michael; Becker, Jan; Bhat, Suhrid; Dahlkamp, Hendrik; Dolgov, Dmitri; Ettinger, Scott; Haehnel, Dirk; Hilden, Tim; Hoffmann, Gabe; Huhnke, Burkhard, et al.: Junior: The stanford entry in the urban challenge, in: Journal of field Robotics, Vol. 25, pp. 569–597, 2008

Montufar, G. F. et al.: On the number of linear regions of deep neural networks (2014)
Montufar, Guido F; Pascanu, Razvan; Cho, Kyunghyun,; Bengio, Yoshua: On the number of linear regions of deep neural networks, in: Advances in neural information processing systems, Vol. 27, 2014

Morris, C. et al.: Voltage oscillations in the barnacle giant muscle fiber (1981)
Morris, Catherine; Lecar, Harold: Voltage oscillations in the barnacle giant muscle fiber, in: Biophysical journal, Vol. 35, pp. 193–213, 1981

Narksri, P. et al.: A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles (2018)
Narksri, Patiphon; Takeuchi, Eijiro; Ninomiya, Yoshiki; Morales, Yoichi; Akai, Naoki,; Kawaguchi, Nobuo: A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles, in: 2018 21st International Conference on intelligent transportation systems (ITSC), IEEE, pp. 497–504, 2018

National Highway Traffic Safety Administration and others: National motor vehicle crash causation survey (2008)
National Highway Traffic Safety Administration and others: National motor vehicle crash causation survey: Report to congress, in: National Highway Traffic Safety Administration Technical Report DOT HS, Vol. 811, p. 059, 2008

Neil, D. et al.: Effective sensor fusion with event-based sensors and deep network architectures (2016)
Neil, Daniel; Liu, Shih-Chii: Effective sensor fusion with event-based sensors and deep network architectures, in: 2016 IEEE international symposium on circuits and systems (ISCAS), IEEE, pp. 2282–2285, 2016

Neubeck, A. et al.: Efficient non-maximum suppression (2006)
Neubeck, Alexander; Van Gool, Luc: Efficient non-maximum suppression, in: 18th international conference on pattern recognition (ICPR'06), IEEE, vol. 3, pp. 850–855, 2006

Oswald, A.-M. M. et al.: Interval coding. I. Burst interspike intervals as indicators of stimulus intensity (2007)
Oswald, Anne-Marie M; Doiron, Brent,; Maler, Leonard: Interval coding. I. Burst interspike intervals as indicators of stimulus intensity, in: Journal of neurophysiology, Vol. 97, pp. 2731–2743, 2007

Paden, B. et al.: A survey of motion planning and control techniques for self-driving urban vehicles (2016)
Paden, Brian; Čáp, Michal; Yong, Sze Zheng; Yershov, Dmitry,; Frazzoli, Emilio: A survey of motion planning and control techniques for self-driving urban vehicles, in: IEEE Transactions on intelligent vehicles, Vol. 1, pp. 33–55, 2016

Pape, L.: Kalifornien stoppt fahrerlose Fahrten mit Cruise-Robotaxis (2023)
Pape, Leonardo: Kalifornien stoppt fahrerlose Fahrten mit Cruise-Robotaxis, URL: https://www.zeit.de/mobilitaet/2023-10/kalifornien-cruise-robotaxi-unfall-erlaubnis, 2023, visited on 11/08/2023

Park, S. et al.: Fast and efficient information transmission with burst spikes in deep spiking neural networks (2019)
Park, Seongsik; Kim, Seijoon; Choe, Hyeokjun,; Yoon, Sungroh: Fast and efficient information transmission with burst spikes in deep spiking neural networks, in: Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1–6, 2019

Paugam-Moisy, H. et al.: Computing with spiking neuron networks. (2012)
Paugam-Moisy, Hélene; Bohte, Sander M: Computing with spiking neuron networks. In: Handbook of natural computing, Vol. 1, pp. 1–47, 2012

Paula Veronese, L. de et al.: Evaluating the limits of a LiDAR for an autonomous driving localization (2020)
Paula Veronese, Lucas de; Auat-Cheein, Fernando; Mutz, Filipe; Oliveira-Santos, Thiago; Guivant, José E; De Aguiar, Edilson; Badue, Claudine,; De Souza, Alberto Ferreira: Evaluating the limits of a LiDAR for an autonomous driving localization, in: IEEE Transactions on Intelligent Transportation Systems, Vol. 22, pp. 1449–1458, 2020

Pehle, C. et al.: The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity (2022)
Pehle, Christian; Billaudelle, Sebastian; Cramer, Benjamin; Kaiser, Jakob; Schreiber, Korbinian; Stradmann, Yannik; Weis, Johannes; Leibfried, Aron; Müller, Eric,; Schemmel, Johannes: The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity, in: Frontiers in Neuroscience, Vol. 16, p. 795876, 2022

Pfeiffer, M. et al.: Deep learning with spiking neurons: Opportunities and challenges (2018)
Pfeiffer, Michael; Pfeil, Thomas: Deep learning with spiking neurons: Opportunities and challenges, in: Frontiers in neuroscience, Vol. 12, p. 774, 2018

Portelli, G. et al.: Rank order coding: a retinal information decoding strategy (2016)
Portelli, Geoffrey; Barrett, John M; Hilgen, Gerrit; Masquelier, Timothée; Maccione, Alessandro; Di Marco, Stefano; Berdondini, Luca; Kornprobst, Pierre,; Sernagor, Evelyne: Rank order coding: a retinal information decoding strategy revealed by large-scale multielectrode array retinal recordings, in: eneuro, Vol. 3, 2016

Qi, C. R. et al.: Pointnet: Deep learning on point sets for 3d classification and segmentation (2017)
Qi, Charles R; Su, Hao; Mo, Kaichun,; Guibas, Leonidas J: Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652–660, 2017

Qi, C. R. et al.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space (2017)
Qi, Charles Ruizhongtai; Yi, Li; Su, Hao,; Guibas, Leonidas J: Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Advances in neural information processing systems, Vol. 30, 2017

Ranganathan, A. et al.: Light-weight localization for vehicles using road markings (2013)
Ranganathan, Ananth; Ilstrup, David,; Wu, Tao: Light-weight localization for vehicles using road markings, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 921–927, 2013

Rasshofer, R. H. et al.: Advances in Radio Science Automotive Radar and Lidar Systems (2005)
Rasshofer, R H; Gresser, K: Advances in Radio Science Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions, in: vol. 3, pp. 205–209, 2005

Redmon, J. et al.: YOLO9000: better, faster, stronger (2017)
Redmon, Joseph; Farhadi, Ali: YOLO9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271, 2017

Ren, S. et al.: Faster r-cnn: Towards real-time object detection with region proposal networks (2015)
Ren, Shaoqing; He, Kaiming; Girshick, Ross,; Sun, Jian: Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, Vol. 28, 2015

Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. (1958)
Rosenblatt, Frank: The perceptron: a probabilistic model for information storage and organization in the brain. In: Psychological review, Vol. 65, p. 386, 1958

Ross, P. E.: The Audi A8: the World's First Production Car to Achieve Level 3 Autonomy (2017)
Ross, Philip E.: The Audi A8: the World's First Production Car to Achieve Level 3 Autonomy, URL: https://spectrum.ieee.org/the-audi-a8-the-worlds-first-production-car-to-achieve-level-3-autonomy, 2017, visited on 07/07/2023

Rueckauer, B. et al.: Conversion of continuous-valued deep networks (2017)
Rueckauer, Bodo; Lungu, Iulia-Alexandra; Hu, Yuhuang; Pfeiffer, Michael,; Liu, Shih-Chii: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification, in: Frontiers in neuroscience, Vol. 11, p. 682, 2017

Rumelhart, D. E. et al.: Learning representations by back-propagating errors (1986)
Rumelhart, David E; Hinton, Geoffrey E,; Williams, Ronald J: Learning representations by back-propagating errors, in: nature, Vol. 323, pp. 533–536, 1986

Rusu, R. B.: Semantic 3D object maps for everyday manipulation in human living environments (2010)
Rusu, Radu Bogdan: Semantic 3D object maps for everyday manipulation in human living environments, in: KI-Künstliche Intelligenz, Vol. 24, pp. 345–348, 2010

SAE international: Taxonomy and definitions for terms related to driving automation systems (2018)
SAE international: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, in: SAE international, Vol. 4970, pp. 1–5, 2018

Savransky, S. D.: Engineering of creativity: Introduction to TRIZ methodology (2000)
Savransky, Semyon D: Engineering of creativity: Introduction to TRIZ methodology of inventive problem solving, CRC press, 2000

Sengupta, A. et al.: Going deeper in spiking neural networks: VGG and residual architectures (2019)
Sengupta, Abhronil; Ye, Yuting; Wang, Robert; Liu, Chiao,; Roy, Kaushik: Going deeper in spiking neural networks: VGG and residual architectures, in: Frontiers in neuroscience, Vol. 13, p. 95, 2019

Simon, M. et al.: Complex-yolo: Real-time 3d object detection on point clouds (2018)
Simon, Martin; Milz, Stefan; Amende, Karl,; Gross, Horst-Michael: Complex-yolo: Real-time 3d object detection on point clouds, in: arXiv preprint arXiv:1803.06199, 2018

Singler, P.: Haftungsprobleme bei autonomen Fahrzeugen (2017)
Singler, Philipp: Haftungsprobleme bei autonomen Fahrzeugen, in: Freilaw: Freiburg L. Students J. P. 14, 2017

Song, S. et al.: Competitive Hebbian learning through spike-timing-dependent synaptic plasticity (2000)
Song, Sen; Miller, Kenneth D,; Abbott, Larry F: Competitive Hebbian learning through spike-timing-dependent synaptic plasticity, in: Nature neuroscience, Vol. 3, pp. 919–926, 2000

Stein, R. B.: Some models of neuronal variability (1967)
Stein, Richard B: Some models of neuronal variability, in: Biophysical journal, Vol. 7, pp. 37–68, 1967

Sun, P. et al.: Scalability in perception for autonomous driving: Waymo open dataset (2020)
Sun, Pei; Kretzschmar, Henrik; Dotiwalla, Xerxes; Chouard, Aurelien; Patnaik, Vijaysai; Tsui, Paul; Guo, James; Zhou, Yin; Chai, Yuning; Caine, Benjamin, et al.: Scalability in perception for autonomous driving: Waymo open dataset, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446–2454, 2020

Tavanaei, A. et al.: Deep learning in spiking neural networks (2019)
Tavanaei, Amirhossein; Ghodrati, Masoud; Kheradpisheh, Saeed Reza; Masquelier, Timothée,; Maida, Anthony: Deep learning in spiking neural networks, in: Neural networks, Vol. 111, pp. 47–63, 2019

Thorpe, S. et al.: Rank order coding (1998)
Thorpe, Simon; Gautrais, Jacques: Rank order coding, in: (Hrsg.): Computational Neuroscience: Trends in Research, 1998, Springer, 1998

Tovee, M. J. et al.: Information encoding and the responses of single neurons in the primate temporal visual cortex (1993)
Tovee, Martin J; Rolls, Edmund T; Treves, Alessandro,; Bellis, Raymond P: Information encoding and the responses of single neurons in the primate temporal visual cortex, in: Journal of neurophysiology, Vol. 70, pp. 640–654, 1993

UN Regulation: No.157 Automated Lane Keeping Systems (ALKS) (2021)
UN Regulation: No.157 Automated Lane Keeping Systems (ALKS), in: Nations Economic Commission for Europe: Geneva, Switzerland, pp. 75–137, 2021

Urmson, C. et al.: Tartan racing: A multi-modal approach to the darpa urban challenge (2007)
Urmson, Chris; Bagnell, J Andrew; Baker, Christopher; Hebert, Martial; Kelly, Alonzo; Rajkumar, Raj; Rybski, Paul E; Scherer, Sebastian; Simmons, Reid; Singh, Sanjiv, et al.: Tartan racing: A multi-modal approach to the darpa urban challenge, 2007

Vicol, A.-D. et al.: Real-time classification of LIDAR data using discrete-time Recurrent Spiking Neural Networks (2022)
Vicol, Anca-Diana; Yin, Bojian,; Bohté, Sander M: Real-time classification of LIDAR data using discrete-time Recurrent Spiking Neural Networks, in: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1–9, 2022

Wallace, A. M. et al.: Full waveform LiDAR for adverse weather conditions (2020)
Wallace, Andrew M; Halimi, Abderrahim,; Buller, Gerald S: Full waveform LiDAR for adverse weather conditions, in: IEEE transactions on vehicular technology, Vol. 69, pp. 7064–7077, 2020

Wang, S. P.: Computer Architecture and Organization: Fundamentals and Architecture Security (2021)
Wang, Shuangbao Paul: Computer Architecture and Organization: Fundamentals and Architecture Security, Springer, 2021

Wang, W. et al.: Temporal pulses driven spiking neural network for time and power efficient object recognition (2021)
Wang, Wei; Zhou, Shibo; Li, Jingxi; Li, Xiaohua; Yuan, Junsong,; Jin, Zhanpeng: Temporal pulses driven spiking neural network for time and power efficient object recognition in autonomous driving, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, pp. 6359–6366, 2021

Wang, W. et al.: Incremental and enhanced scanline-based segmentation method (2016)
Wang, Weimin; Sakurada, Ken,; Kawaguchi, Nobuo: Incremental and enhanced scanline-based segmentation method for surface reconstruction of sparse LiDAR data, in: Remote Sensing, Vol. 8, p. 967, 2016

Wang, Y. et al.: Multi-modal 3d object detection in autonomous driving: a survey (2023)
Wang, Yingjie; Mao, Qiuyu; Zhu, Hanqi; Deng, Jiajun; Zhang, Yu; Ji, Jianmin; Li, Houqiang,; Zhang, Yanyong: Multi-modal 3d object detection in autonomous driving: a survey, in: International Journal of Computer Vision, pp. 1–31, 2023

Waymo LLC: Seeing the road ahead (2023)
Waymo LLC: Seeing the road ahead, URL: https://waymo.com/about/, 2023, visited on 11/08/2023

Weste, N. H.; Eshraghian, K.: Principles of CMOS VLSI design: a systems perspective (1985)
Weste, Neil HE; Eshraghian, Kamran: Principles of CMOS VLSI design: a systems perspective, Addison-Wesley Longman Publishing Co., Inc., 1985

Widrow, B. et al.: Adaptive switching circuits (1960)
Widrow, Bernard; Hoff, Marcian E, et al.: Adaptive switching circuits, in: IRE WESCON convention record, New York, vol. 4, pp. 96–104, 1960

Wu, Y. et al.: Deep convolutional neural network with independent softmax for large scale face recognition (2016)
Wu, Yue; Li, Jun; Kong, Yu,; Fu, Yun: Deep convolutional neural network with independent softmax for large scale face recognition, in: Proceedings of the 24th ACM international conference on Multimedia, pp. 1063–1067, 2016

Wu, Y. et al.: Spatio-temporal backpropagation for training high-performance spiking neural networks (2018)
Wu, Yujie; Deng, Lei; Li, Guoqi; Zhu, Jun,; Shi, Luping: Spatio-temporal backpropagation for training high-performance spiking neural networks, in: Frontiers in neuroscience, Vol. 12, p. 331, 2018

Wu, Y. et al.: Direct training for spiking neural networks: Faster, larger, better (2019)
Wu, Yujie; Deng, Lei; Li, Guoqi; Zhu, Jun; Xie, Yuan,; Shi, Luping: Direct training for spiking neural networks: Faster, larger, better, in: Proceedings of the AAAI conference on artificial intelligence, vol. 33, pp. 1311–1318, 2019

Xie, J. et al.: Semantic instance annotation of street scenes by 3d to 2d label transfer (2016)
Xie, Jun; Kiefel, Martin; Sun, Ming-Ting,; Geiger, Andreas: Semantic instance annotation of street scenes by 3d to 2d label transfer, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 3688–3697, 2016

Xu, Y. et al.: A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks (2013)
Xu, Yan; Zeng, Xiaoqin; Han, Lixin,; Yang, Jing: A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks, in: Neural Networks, Vol. 43, pp. 99–113, 2013

Yan, Y. et al.: Second: Sparsely embedded convolutional detection (2018)
Yan, Yan; Mao, Yuxing,; Li, Bo: Second: Sparsely embedded convolutional detection, in: Sensors, Vol. 18, p. 3337, 2018

Yoo, H. W. et al.: MEMS-based lidar for autonomous driving (2018)
Yoo, Han Woong; Druml, Norbert; Brunner, David; Schwarzl, Christian; Thurner, Thomas; Hennecke, Marcus,; Schitter, Georg: MEMS-based lidar for autonomous driving, in: e & i Elektrotechnik und Informationstechnik, 2018

Yurtsever, E. et al.: A survey of autonomous driving: Common practices and emerging technologies (2020)
Yurtsever, Ekim; Lambert, Jacob; Carballo, Alexander,; Takeda, Kazuya: A survey of autonomous driving: Common practices and emerging technologies, in: IEEE access, Vol. 8, pp. 58443–58469, 2020

Zeldenrust, F. et al.: Neural coding with bursts—current state and future perspectives (2018)
Zeldenrust, Fleur; Wadman, Wytse J,; Englitz, Bernhard: Neural coding with bursts—current state and future perspectives, in: Frontiers in computational neuroscience, Vol. 12, p. 48, 2018

Zhang, M. et al.: Efficient spiking neural networks with logarithmic temporal coding (2020)
Zhang, Ming; Gu, Zonghua; Zheng, Nenggan; Ma, De,; Pan, Gang: Efficient spiking neural networks with logarithmic temporal coding, in: IEEE Access, Vol. 8, pp. 98156–98167, 2020

Zhao, Z.-Q. et al.: Object detection with deep learning: A review (2019)
Zhao, Zhong-Qiu; Zheng, Peng; Xu, Shou-tao,; Wu, Xindong: Object detection with deep learning: A review, in: IEEE transactions on neural networks and learning systems, Vol. 30, pp. 3212–3232, 2019

Zhou, S. et al.: Deep SCNN-Based real-time object detection for self-driving vehicles using LiDAR temporal data (2020)
Zhou, Shibo; Chen, Ying; Li, Xiaohua,; Sanyal, Arindam: Deep SCNN-Based real-time object detection for self-driving vehicles using LiDAR temporal data, in: IEEE Access, Vol. 8, 2020

Zhou, S. et al.: Object Detection based on LIDAR Temporal Pulses using Spiking Neural Networks (2018)
Zhou, Shibo; Wang, Wei: Object Detection based on LIDAR Temporal Pulses using Spiking Neural Networks, 2018

Zhou, Y. et al.: Voxelnet: End-to-end learning for point cloud based 3d object detection (2018)
Zhou, Yin; Tuzel, Oncel: Voxelnet: End-to-end learning for point cloud based 3d object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4490–4499, 2018

Zwicky, F.: Entdecken, erfinden, forschen im morphologischen Weltbild (1966)
Zwicky, Fritz: Entdecken, erfinden, forschen im morphologischen Weltbild, in: München: Droemer, 1966