TU WIEN Informatics

Attentional Neural Network based Dynamic Object Detection for Autonomous Multi-Agent Systems

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Daniel Scheuchenstuhl, BSc
Matrikelnummer 01630368

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Mitwirkung: Univ.Ass. Dott.mag. Luigi Berducci

Wien, 12. August 2023

Daniel Scheuchenstuhl	Radu Grosu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Attentional Neural Network based Dynamic Object Detection for Autonomous Multi-Agent Systems

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Daniel Scheuchenstuhl, BSc
Registration Number 01630368

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Assistance: Univ.Ass. Dott.mag. Luigi Berducci

Vienna, 12th August, 2023

Daniel Scheuchenstuhl	Radu Grosu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Daniel Scheuchenstuhl, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. August 2023

Daniel Scheuchenstuhl

Danksagung

Erstmals möchte ich meinen Betreuer Luigi Berducci für die kontinuierliche und umfassende Unterstützung im Rahmen dieser Arbeit honorieren. Zudem möchte ich Jie He für sein wertvolles Feedback und sein Engagement im Bezug auf die Korrekturlesung dieser Arbeit danken.

Des Weiteren möchte ich die Gelegenheit nutzen, mich bei meiner Familie für die tatkräftige Ermutigung und die finanzielle Unterstützung im Rahmen meines gesamten Studiums zu bedanken. Schlussendlich möchte ich den emotionalen Beistand meiner Freunde und meines Freundes würdigen.

Acknowledgements

Above all, I would like to honor my advisor Luigi Berducci for his continuous and comprehensive support throughout this thesis. I would also like to express my gratitude to Jie He for his valuable feedback and commitement proofreading this work.

Furthermore, I would like to take this opportunity to thank my family for their energetic encouragement and financial aid throughout my study. Finally, I would like to show my appreciation for the emotional support of my friends and my boyfriend.

Kurzfassung

Das Erlernen robuster Merkmalsrepräsentationen bleibt ein anspruchsvolles Problem in der Robotik, insbesondere in Anbetracht komplexer visueller Eingaben. Inspiriert durch den menschlichen Aufmerksamkeitsmechanismus, welcher es Menschen ermöglicht, komplexe visuelle Szenen schnell zu verarbeiten und auf Umweltreize zu reagieren, zeigen wir, dass wir durch Einbetten von menschlichen Aufmerksamkeitsmerkmalen in Objekterkennungsalgorithmen die Effizienz und Robustheit der Objekterkennungsalgorithmen verbessern können. In dieser Masterarbeit präsentieren wir eine neuartige Methode zur Emulation von menschlicher Aufmerksamkeit mit einem approximierten maschinellen Lernmodell, indem wir auf menschlichen Blickaufzeichnungen lernen, die während des manuellen Fahrens in einer realen Umgebung im kleinen Maßstab aufgezeichnet wurden. Dementsprechend nutzen wir die gelernten menschlichen Aufmerksamkeitsmerkmale, um die visuellen Eingaben des Objekterkennungsmodells zu bereichern. Die in dieser Arbeit durchgeführten Experimente vergleichen unseren Ansatz mit modernen Objekterkennungsmethoden im Bereich des maschinellen Sehens und zeigen, dass die Nutzung vorhergesagter menschlicher Aufmerksamkeit zu einer verbesserten Robustheit der trainierten Objekterkennungsmodelle in Szenarien außerhalb der Trainingsverteilung führt. Damit betont diese Arbeit das Potenzial der Integration von zusätzlichen menschlichen Aufmerksamkeitsmerkmalen im Rahmen des Repräsentationslernens für die Robotik und zeigt neue Wege für zukünftige Forschungsrichtungen auf.

Abstract

Learning robust feature representations remains a challenging problem in robotics, especially when considering complex visual inputs. Inspired by the human-attention mechanism, allowing humans to rapidly process complex visual scenes and react to environmental stimuli, we show that by embedding human-attention feature maps into object detection pipelines, we can enhance the efficiency and robustness of the object detection algorithms. In this master thesis, we present a novel method for emulating human-attention with an approximated Machine Learning (ML) model by learning on human-gaze recordings of manual driving in a small-scale real-world setting. Consequently, we exploit the learned human-attention feature maps, enriching the visual inputs of the object detection model. The experiments conducted in this thesis compare our approach to state-of-the-art computer vision-based object detection baselines and demonstrate that leveraging predicted human-attention results in improved robustness of the trained object detection models on Out-of-Distribution (OOD) scenarios. To that end, this work emphasizes the potential of integrating auxiliary human-attention features in representation learning for robotics and illustrates new avenues for future research directions.

Contents

CHAPTER 1

Introduction

Over the past few years, computer vision and the advent of Artificial Intelligence (AI) have dramatically advanced and shaped myriad industries and research fields. Among these advancements is autonomous driving, which heavily relies on computer vision systems that are geared toward perceiving and interpreting their surroundings. With remarkable progress in the development of object detection models, autonomous vehicles can now identify and track objects in real-time, enhancing decision-making and enabling safer and more sophisticated autonomous driving systems. However, despite significant progress, OOD scenarios such as altering lighting conditions still pose a tremendous challenge to recent object detection models. Illumination perturbations which are caused by factors such as changing seasons, weather conditions or varying times of day, can severely impact the accuracy and robustness of object detection systems in autonomous driving scenarios.

The accurate detection and recognition of objects in images or videos heavily depends on the availability of distinct visual features and the expressiveness of the learned abstract latent feature representations. Illumination perturbations, characterized by variations in lighting, shadows and contrast, introduce significant challenges in detecting objects precisely, leading to potential hazards in autonomous driving systems. Consequently, robustifying computer vision-based object detection towards illumination perturbations becomes critical for developing reliable and robust autonomous driving systems. Traditional approaches usually apply image enhancement techniques or data augmentation in terms of adjusting brightness levels which often fail to fully address these perturbations, due to the absent incorporation of the underlying perceptual cues utilized by humans.

Recognizing the potential of human attention as a guiding mechanism for object detection, this master thesis aims to explore the benefits of incorporating human attention mechanisms to tackle the issue of illumination perturbations in object detection models for autonomous driving. As an integral component of human visual perception, human attention enables us to selectively focus on relevant objects or regions in our visual Field

of View (FoV) while filtering out irrelevant information. By leveraging this innate human ability, we can develop more robust and adaptable object detection models capable of mitigating illumination perturbations encountered in autonomous driving scenarios.

This research endeavors to contribute to the existing knowledge base in the fields of computer vision, AI and autonomous driving by investigating novel approaches that combine object detection models with the biological human-attention mechanism. For the purpose of creating an autonomous system, a human-attention model mimicking human-attention in driving situations is designed in order to remove human interaction. The insights gained from this study will inform the development of more robust and reliable algorithms for object detection under varying lighting conditions. Finally, by integrating human-attention, this work aims to enhance the safety, robustness and effectiveness of autonomous driving systems, enabling them to detect and track objects under varying lighting conditions and challenging environments more accurately. To this end, this thesis aims to answer the following research questions:

RQ0: How can we obtain a rich enough labeled dataset for human attention in driving situations?

RQ1: How to design a neural network capable of imitating human attention for object detection in autonomous driving?

RQ2: What is the comparative performance of state-of-the-art object detection approaches, their suitability in our context, and the optimal overall architecture for an attentional object detector?

The subsequent chapters will delve into the related work, the scientific background and the research methodology employed in this thesis. Afterwards, we will demonstrate a novel approach for object detection utilizing imitated human-attention to improve the robustness of our object detection algorithm. The main contributions of our research are provided by an in-depth comparison of state-of-the-art object detection models with our system on the impact of illumination perturbations on their performance. This work is evaluated on a F1/10 scale setup whereby the human-attention model as well as all object detection models are trained and evaluated on custom datasets. The F1Tenth research platform [1] is used along eye-tracking glasses to enable the recording of human-attention and the acquisition of the necessary datasets for a proof-of-concept evaluation in a small-scale setup. While most recent object detection approaches utilizing attention mechanisms apply artificial attention as auxiliary feature [2, 3], the Detection Transformer (DETR) [4] harnesses the power of the transformer architecture, primarily relying on its artificial self-attention mechanism for object detection. For this purpose, we further provide an exemplary comparison on the object detection results, our predicted human-attention feature maps and the artificial self-attention utilized by the transformer architecture of DETR. Then, we reason about the underlying mechanics of the imitated

human-attention and the artificial attention based on the given example respectively. Concluding our evaluation, we used the F1Tenth research platform to conduct a feasibility study of our approach for real-world autonomous driving/racing systems.

Overall, this thesis aims to contribute to the advancement of object detection models for autonomous driving by addressing the critical issue of illumination perturbations through the utilization of predicted human-attention. By enhancing the adaptability and robustness of these systems, this work endeavors to advance the state-of-the-art in this field, paving the way for safer and more reliable autonomous vehicles in the future.

CHAPTER 2

Scientific Background

According to the state-of-the-art on this research topic and the stated problem description in Chapter 1, this chapter presents the state-of-the-art approaches in the visual object detection task. Furthermore, this chapter intends on providing a principle overview of the scientific work that has already been done on developing computer vision systems, specifically object recognition and object detection approaches incorporating some basic form of self-attention mechanism. Moreover, relevant scientific work that has been done on similar research topics as well as the technical background this thesis builds on is discussed. For this reason, the first section describes the related work while the second section and onward provide a general introduction on the technical background of this thesis.

2.1 Related Works

In this section, we state the works related to the contributions of this thesis. Considering the application domains of object detection and autonomous racing, we conducted a thorough review of the existing work of learning-based approaches used in autonomous racing focusing on the perception end of the perception, planning and control pipeline applied in robotics.

Human-Attention Models. Inspired by it's vast impact on human learning and perception, human attention has been comprehensively studied for almost a century. Johnson et. al. [5] extensively discuss the fundamental mechanism of human attention and its benefit for enhancing human learning. Focusing on human visual perception, Tsotsos [6] analyzes vision at the complexity level and refines the computational nature of the visual search task. Moreover, Itti et. al. [7] even provide a framework for a computational and neurobiological explanation of human visual attention. Recent research demonstrates that the notion of attentional focus can efficiently be adopted to enhance the performance of various ML applications such as Natural Language Processing (NLP) [8], machine

translation [9] or object detection [3]. However, the relation between machine attention utilized by modern Neural Network (NN)s and human attention remains unresolved. For this purpose, Sood et. al. [10] analyze and interpret the focus of NNs in comparison to human visual attention in the context of a machine reading comprehension. To this end, we demonstrate the practical use of human visual attention based feature maps to enhance the performance of computer vision models.

Representation Learning. Significant progress in the field of representation learning has been achieved in recent years [11]. In order to reach the goal of learning low-dimensional, latent feature representations, most modern ML approaches either rely on self-supervised learning or generative adversarial networks [12, 13, 14] using autoencoders [15, 16, 17]. Alternatively, Moyer et. al. [18] show that adversarial training is not always optimal for learning invariant feature representations. In the context of self-supervised learning, recently emerged contrastive learning algorithms achieve superior results for learning strong visual representations [19, 20]. When applied to small models, contrastive learning may only yield poor performance. To this end, Fang et. al. [21] leverage the visual representations learned by a larger network for a smaller model using self-supervised learning. In contrast to previous research, we do not aim to use human attention to derive efficient feature representations. Instead, we propose to enrich the learned visual feature representations with human visual attention based feature maps.

Autonomous Racing. Given that our robotics experiments have been framed in the context of autonomous racing, we include a review of existing approaches focusing on deep-learning based methods w.r.t. perception. A complete overview of the research field of autonomous racing is given in [22]. While only few works tackle object detection using multi-modal inputs and sensor fusion [23, 24, 25], most approaches aim to improve the performance of existing visual object detection algorithms [26, 27]. In particular, the You Only Look Once (YOLO) architecture [28] has seen extensive advancements regarding object detection accuracy, inference time and latency in recent years. Despite the impressive results of it's latest successor [29], the model is still sensitive to rapid variations in illumination. Comparably to [24, 25], our goal is to improve the robustness of object detection architectures to changes in illumination via additional sensory information, namely human-attention based feature maps.

Object Detection. Object detection is one of the fundamental problems of computer vision. With the rise of AI and ML, a vast amount of techniques and methods based on Deep Neural Network (DNN)s and Convolutional Neural Network (CNN)s have been proposed such as [30] [31]. Most state-of-the-art object detection algorithms treat object detection as a regression problem, where classification and localization of the objects in the image are performed simultaneously. In general, state-of-the-art object detection algorithms can be categorized into one-stage methods such as the YOLO family [28, 32, 33, 34, 35, 36, 29] and Single Shot Detector (SSD) [37] and two-stage methods such as Regions with CNN features (R-CNN) [38], Fast/Faster R-CNN [39, 40] and R-FCN [41] depending on whether region proposals are computed a priori. While two-stage methods provide a higher object detection accuracy, one-stage methods are

superior in inference time. Moreover, Lin et. al. [42] proposed a simple and powerful framework for building feature pyramid networks to be used inside CNNs for an efficient generation of region proposals.

Based on the major advances in NLP, document summarization and machine translation tasks by implementing the concept of self-attention following [8], attention has also been considered and successfully applied in computer vision tasks such as object detection [3, 2]. Specifically, Hara et. al. [3] proposed an augmenting deep neural network with an attention mechanism for visual object detection. When compared to Fast R-CNN [39], a consistent performance improvement of the self-attention based DNNs can be determined. Furthermore, Zhu et. al. [2] proposed a fully convolutional network denoted as Attention CoupleNet to incorporate the attention-related global and local information of objects to improve the object detection performance. Attention CoupleNet achieves state-of-the-art performance on the Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes (PASCAL VOC) and Common Objects in Context (COCO) datasets for object detection. Moreover, the Transformer architecture proposed by Vaswani et al. [8] has also been successfully applied to computer vision applications such as image recognition [43] and object detection [4]. Both approaches achieve competitive results in their respective domain of application.

However, all of the discussed methods that incorporate an attention mechanism implement the concept of self-attention following [8] while a more advanced attention concept imitating human attention may proof beneficial in terms of training time, object detection accuracy, inference time or robustness [44]. Ultimately, for most object detection approaches, the PASCAL VOC and COCO benchmarks are the defacto standard in object detection performance comparison while they cannot be used as measure for quantifying the object detection performance in an autonomous driving context.

2.2 Human Attention

The selective focus on specific regions of a visual scene for fast perception is biologically integrated in the human visual system and known as human attention [44]. Alternatively, human attention may be described as a procedure for reducing the computational cost of the search process inherent to visual perception [7]. Based on the rising computational complexity of computer vision tasks, data-driven visual search approaches become infeasible addressing the need for an attention based feature selection method that is further able to exploit and use task knowledge in order to minimize the computational cost of visual processing [6]. Due to the nature of human visual attention, an attention based visual search algorithm that efficiently mimics human visual attention is desired. According to [7], four general computer vision approaches have emerged: artificial manipulations, active vision, perceptual grouping and Regions of Interest (RoI) operators. Artificial manipulations are highly task-specific systems requiring domain-knowledge and a variety of assumptions. Furthermore, these systems do not only rely on vision and do not generalize well due to their task-specific design. In active vision, features

are extracted from a visual scene by manipulating the parameters of the sensors. For example when applied in structure reconstruction, stereo camera systems are used to reconstruct a visual scene based on the variation of the degrees of freedom of the sensors (position, focus, zoom, vergence, etc.). Another idea is to cluster/organize low-level features into high-level structures to obtain a more abstract view of the visual scene allowing for semantic manipulation of the segmented scene. While such generic methods are not specifically designed for computer vision, in a biological and psychological context such processes are also known as perceptual grouping algorithms which are constrained by the principles of Gestalt theory. Last but not least, RoI operators specify distinctive and invariant descriptors of features of interest in a scene. Hence, each feature is uniquely characterized with respect to its neighbouring features in terms of position, geometry and radiometric distortions. Although the selective processing of RoI leads to promising results, the lack of semantic information in the low-level feature representations limit its general applicability.

In order to create a biological model of visual attention, empirical research must be conducted and experimental data must be obtained leading to the study of visual search, saliency, overt and covert attention as well as bottom-up/top-down attention. Visual search deals with the localization of a target items among several distraction items where either parallel or serial visual search is applied. The reaction time in parallel visual search is independent of the number of distraction items whereas the reaction time in serial visual search increases proportional to the number of additional distraction items at a given rate. It is assumed that in parallel visual search, all items are processed simultaneously at a semantic level sufficient to distinct between target and distraction items while this distinction is not possible anymore in serial visual search. Another key aspect of visual attention is saliency which is biologically implemented as local saliency modulation and global saliency map. Local saliency modulation is assumed to be performed in the striate cortex and interpreted as precursor or texture/object segregation. Contrarily, a global saliency map is assumed to be located in the lateral intraparietal (LIP) area and represents salient stimuli as the neurons in this region seem to respond very well to recently flashed stimuli and abrupt motion. In a computational modelling context, a global saliency map can basically be modeled as 2D activation map (heatmap) of varying neuron firing intensities. Coming back to attention in general, we can distinguish between overt attention and covert attention. By definition, overt attention is the fixation or direct focus of the eyes on a respective stimuli in the visual perception. On the other hand, covert attention is described by reacting on a stimuli that is located within the radial bounds of peripheral vision but not fixated by the gaze of the eyes. Covert attention is also denoted as visual spatial attention and initiates the transition of overt attention in the attentional locus. Finally, we can discuss the impact of bottom-up and top-down attention. Stimulus-related attention or also denoted as bottom-up attention is observed by differentiating between various items and shifting the attention in terms of the fixation of the eyes to the most relevant stimuli. Top-down attention is driven by an intention in terms of searching the visual perception for specific information. This information may either be explicitly present e.g. image cues or implicitly provided via prior stimuli.

Based on research conducted in this field, three of the major theories of visual attention shall be briefly introduced. The most popular theory of visual attention is attention as selection which views visual attention as mechanism to selectively identify RoI in the visual perception and prepare those for further processing in higher cortical areas. As the human visual system appears to be a hierarchical system that incorporates scale and location invariance in order to process input images, a common theory is attention as filtering. This theory relies on the increasing receptive field size of the neurons at higher levels of the hierarchy where attention is supposed to be used to filter the relevant stimuli from the redundant information and improve the signal-to-noise ratio. The third major theory denoted as attention as process assigns the observed characteristics to emergent properties of the competitive neural network in the brain rather than to a dedicated attentional mechanism.

2.3 Artificial Intelligence & Machine Learning

AI is a field of computer science and refers to the capability of a computer system to imitate human intelligence, develop cognitive abilities or even go beyond human capabilities. Computer systems that incorporate AI have the ability to learn, gain problem-solving competence and evolve. Generally, an AI does not need to be pre-programmed. AI uses mathematical operations and logic to mimic human reasoning in order to arrive at a well-founded decision for a specific problem or task. On the contrary, ML is a subset of AI where a computer system makes a decision or prediction based on historical data. While computer systems incorporating ML do not need to be explicitly pre-programmed either, these systems generally require a vast amount of historical data such that high-confidence predictions on new data can be made. Basically, a ML algorithm constructs a statistical model based on the historical data, also called training data. This model is further evaluated on the validation data to guarantee that the model learns patterns and regularities in the training data and generalizes well on new data instead of overfitting on the training data. In contrast to general AI, ML algorithms are specialized on a specific domain or even a specific problem in particular. In order to effectively learn from the historical data, ML algorithms are modeled based on the knowledge and nature of the human brain.

In more detail, artificial neurons are the elementary units of a ML algorithm and are inspired by the biological neurons in a living organism. Artificial neurons are an abstract representation modeling the information flow of a biological neuron where an input vector is basically summed up to produce an output. Artificial neurons are designed to imitate the computation process that occurs in a biological neuron which is composed of dendrites, soma and axon. Specifically, an artificial neuron receives an input sequence x as input where each part of this input sequence x denoted as x_i is separately multiplied with a specific weight denoted as w_i. The weighted input sequence is then passed on to a transfer function. Usually, the transfer function Σ computes the sum or the dot product of the weighted input sequence and adds a bias term b which is further provided to an activation function as input. Based on the provided input, the implementation of the

activation function ϕ and the threshold value θ, the neuron either activates and passes on its output in terms of a real number or does not activate. Mathematically, the output y can be formulated as a function of input x as follows: $y = \phi(\Sigma x_i * w_i + b)$. Figure 2.1 also illustrates the input-output mapping as well as the information flow of an artificial neuron.

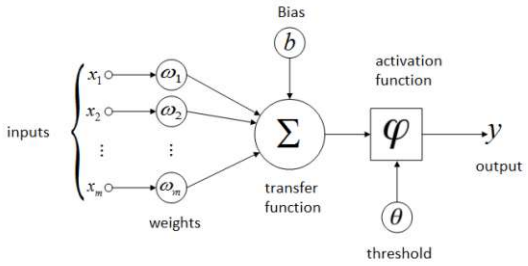


Figure 2.1: Schematic representation of an artificial neuron

Based on the synapses and biological neurons in the nerve system of a living creature that form a self adaptive biological neural network, artificial neurons are similarly connected to each other forming an artificial NN. Artificial NNs are composed of multiple artificial neurons and are organized in a hierarchical structure meaning that the initial input sequence is processed by the first couple of neurons while the output of these neurons serves as input for the next couple of neurons. Information between these groups of neurons is passed on in an unidirectional way until the last group of neurons has received the data and finished processing. The group of neurons that belongs to the same hierarchy in the network is denoted as layer. The most simplistic artificial NNs consist of two layers: the first layer is called the input layer and the last layer is called the output layer. Commonly, the structure of an artificial NN implements a hierarchy with many more layers in between than an input and output layer. These layers are referred to as hidden layers as their outputs are not visible. Generally, the integration of hidden layers in a ML model allows to train an artificial NN for more complex tasks. A hidden layer receives the output of its previous layer and performs a non-linear transformation on the input vector which is further passed on to the next layer. As every hidden layer has different parameters depending on its functional characteristics, distinct transformations are applied to the input. Thus, hidden layers allow the functionality of the artificial NN to be broken down into various functions which can be specifically designed to apply an intended transformation at a given processing step in the data flow to achieve an expected result. For instance, by using a sigmoid activation function in the output layer, the output of the neural network can be transformed into the range between 0 and 1 which might be the intended output of the neural network if a probability is expected.

This conceptual design of artificial NN where multiple hidden layers are stacked to build the structure of the artificial NN is also referred to as DNNs where deep learning corresponds to the overall term of training, designing and validating such networks. Figure 2.2 visualizes the difference between a simple artificial NN and a DNN. The major

advantage of DNNs is that information processing happens at each stage/layer separately meaning that the first couple of hidden layers may extract some low-level information from the training data whereas succeeding hidden layers do not consider all information, but only the relevant information passed on from the preceding hidden layers. This allows deeper hidden layers of the DNN to learn more high-level, general and abstract information about the data while only considering relevant data.

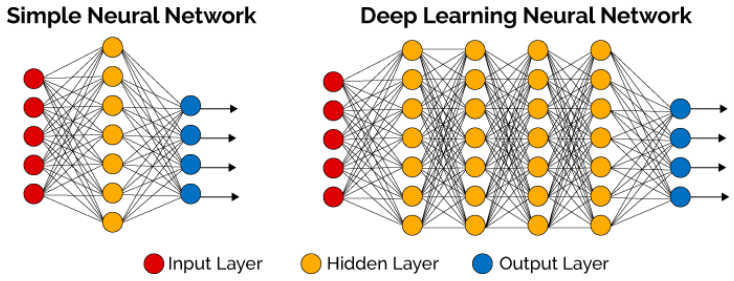


Figure 2.2: Structure of a simple and deep artificial neural network from [45]

The structure of a DNN or also called algorithm of the DNN is highly task specific. For example, a Multilayer Perceptron (MLP) consists of three or more fully-connected layers meaning that every neuron of the preceding layer is connected to every neuron of the succeeding layer given a respective weight and implements non-linear activation functions. Although the MLP can distinguish patterns in non-linearly separable data, it will not be able to efficiently and accurately learn how to classify various images of animals as the MLP can not learn about the spatial information of the data. In order to implement a DNN capable of learning how to classify various images of animals, a variant of a CNN may be used. In CNNs, the key to obtain spatial information between the pixels in an image is by performing a convolution operation on the 2D or 3D image data using a convolutional matrix denoted as kernel filter of predefined weights. By applying consecutive convolution operations in the hidden layers, different convolutional transformations are learned allowing the CNN to extract various spatial features at every layer. The CNN learns about low-level features of the input images such as edges in the upper hidden layers, mid-level features such as corners or contours in the next couple of hidden layers and high-level features such as geometric shapes or facial characteristics in the lower hidden layers. Finally, classification of the abstract feature representations is performed in a couple of fully-connected layers. Inspired by the receptive field of biological neurons, artificial neurons of a convolutional layer have only a local view of the input matrix received from the previous convolutional layer. Additionally, all neurons within a convolutional layer share the same weights allowing them to learn local features. As the exact position of low-level features such as edges is not of interest for object recognition, each convolution layer may be followed by a pooling layer. There are several types of pooling operations including average-pooling and max-pooling while max-pooling is the most commonly used one. For instance, by performing a max-pooling operation on a 2x2 grid, only the neuron from the previous convolution layer with the highest activation in this area is considered for further processing while the outputs of the other three neurons

are discarded. Based on a 2x2 grid, the output space of the previous convolution layer is reduced by a factor of four using a succeeding pooling layer. Thus, pooling layers allow to implement and train deeper NN while also providing a measure to prevent the NN from overfitting. Generally, CNNs vastly reduce the number of parameters required for training in comparison to MLP determining the success of CNNs for image processing tasks.

2.3.1 Activation Functions

An artificial neural network may be viewed as an universal function approximator allowing to learn any function given a specific input-output mapping. The complexity and learning capability of an artificial NN depends on the mathematical model of its artificial neurons. When modeling an artificial neuron merely as a composition of linear functions, the artificial neuron will lack the ability to learn non-linear correlations. Therefore, an artificial NN or DNN which is only composed of a combination of linear transformations can not benefit from the deep learning approach of a multi-layer network as every combination of linear functions has an equivalent single-layer network and can be functionally reduced to it. In order to achieve the described functionality of the activation function and for the artificial neuron to be able to also learn non-linear functions, activation functions have to be modeled as non-linear functions. There are several types of activation functions such as sigmoidal shaped functions, piecewise linear functions or step functions each having different properties regarding network complexity and learning convergence. Common activation functions include the sigmoid function, hyberbolic tangent function, heaviside function as well as Rectified Linear Unit (ReLU) activation function and its variants.

The sigmoid activation function given by $\sigma(x) = \frac{1}{1+e^{-x}}$ is a non-linear and continuous function which is commonly used to normalize any input in the range between 0 and 1. It also comes with an easily calculated derivative which is especially important in gradient-based learning methods such as supervised learning 2.3.4 for computing the gradients in the backpropagation algorithm. On the downside, optimization of multi-layer neural networks becomes a challenge as the gradients tend to diminish towards zero for neurons incorporating a bounded activation function as a sigmoid activation function making the network vulnerable to the vanishing gradient problem. The vanishing gradient problem occurs in gradient-based learning methods during the backpropagation algorithm where the parameters (weights & bias) of the neurons are updated based on the partial derivatives of the loss function with respect to these parameters. These update values are also denoted as gradients. For very deep multi-layer networks, the gradients are successively multiplied based on the chain rule over and over again resulting in insignificantly small update values in upper layers. Thus, if the output of one of the upper layer's neurons is close to a threshold value, the small gradients will prevent the weights from updating their value. Hence, the network either converges really slow or is prevented from learning at all in the worst case.

The hyperbolic tangent function given by $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ is a zero-centered, non-linear and continuous function similar to the sigmoid function where real input values

are suppressed to the range between -1 and 1. In contrast to the sigmoid function, the gradient of the TanH function is much higher at zero where data is usually centered around. Additionally, TanH is symmetric around zero resulting in a much faster convergence. On the downside, the TanH activation function also suffers from the vanishing gradient problem. The heaviside function or also denoted as step function is a non-linear function used for a binary classification of the input given a specific threshold θ. It is usually useful in the last layer of a NN where a binary decision has to be performed. The heaviside function can be approximated using any sigmoidal function.

One of the most commonly used activation functions is the ReLU activation function. ReLU is a piecewise linear and unbound function which corresponds to the positive part of its argument. Mathematically, ReLU is given by $f(x) = max(0, x)$ where x is the input of a neuron. The main benefits of ReLU are its efficient computation as only simple mathematical operations are required in comparison to the sigmoid or TanH function that require an exponential calculation, its representational sparsity allowing for true zero values and accelerating the overall learning process as well as it avoids the vanishing gradient problem allowing to effectively train DNNs. Moreover, ReLU equals the linear ramp function for $x > 0$ making it easier to optimize. Contrarily, based on its definition, ReLU may output any value in the range $[0, \infty)$ possibly leading to the exploding gradient problem which is the compliment of the vanishing gradient problem. The exploding gradient problem occurs when large magnitudes of gradients accumulate and result in very large updates to the weights of a NN. These large updates may cause the NN to become unstable and diverge, making further learning infeasible. The exploding gradient problem may be identified in terms of Not a Number (NaN) values in the training process, as large updates may lead to an overflow in the loss and/or weights. There are a couple of methods in order to fix this issue such as gradient clipping or weight regularization. Revisiting ReLU, the gradients will be zero for activations where the input is in the range $x < 0$, thus preventing the neurons from applying any future updates. This problem is also denoted as the dying ReLU problem. Especially for designing and training DNNs, ReLU has become the default activation function. Therefore, there are many variants of ReLU such as leaky ReLU and Exponential Linear Unit (ELU) each having different properties in order to tackle the drawbacks of the ReLU activation function. For example, the leaky ReLU function given by

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise} \end{cases}$$

is an extension of ReLU and tackles the dying ReLU problem by allowing a small, constant and non-zero gradient for $x < 0$. On the other hand, the ELU given by

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha * (e^x - 1) & \text{otherwise} \end{cases}$$

extends ReLU and attempts to compensate for the dying ReLU problem by smoothly declining its output value until its equal to $-\alpha$ which represents a tunable hyper-parameter.

2.3.2 Structure of Neural Networks

Up to this point, when describing the design and behaviour of NN, we implicitly talked about Feed-Forward Neural Network (FFNN). A FFNN can be modeled as a directed, acyclic graph where information may only flow in the forward direction. Thus, when data is received at the input layer, it is processed and passed on to the first hidden layer where it is again processed and passed on to the next hidden layer. This NN design implies that there is no backward flow which may also be denoted as feedback loop. On the contrary, a Recurrent Neural Network (RNN) is a directed but cyclic graph where information is also passed along a feedback loop. Figure 2.3 shows the structural difference between a RNN and a FFNN.

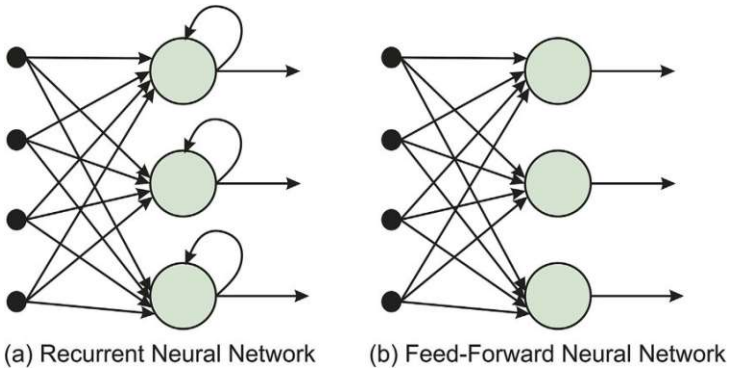


Figure 2.3: Design of a Recurrent Neural Network (RNN) and a Feed-Forward Neural Network (FFNN) [46]

The idea behind RNNs is to adequately learn sequential or time series data such as language translation, NLP or speech recognition. Specifically, the assumption for the design of RNNs is that input and output are not independent of each other but the output depends on the prior input. Hence, for predicting the next token of a sequence of dependant data items such as predicting the next character or word of a sentence, the previous state of the neurons has to be memorized. At each time step, a neuron receives the output of the previous layer as well as its previous output state as input. The short-term memory capabilities of a RNN may be extended to more than only considering one time step ahead though, meaning the RNN can be composed of many hidden units. In order to train a RNN, the Backpropagation Through Time (BPTT) algorithm is applied which is a variant of the backpropagation algorithm where the error is propagated from the last time step to the first time step as the partial error of a given time step depends on the error of the previous time step. Figure 2.4 demonstrates the concept of unrolling the time steps of a RNN.

The major issue of RNNs is that they are also vulnerable to the vanishing gradient problem. For this purpose, Sepp Hochreiter and Juergen Schmidhuber proposed the concept of Long-Short Term Memory (LSTM) [47] which is an extension of RNNs and partially solves the vanishing gradient problem as well as also allows to learn long-term

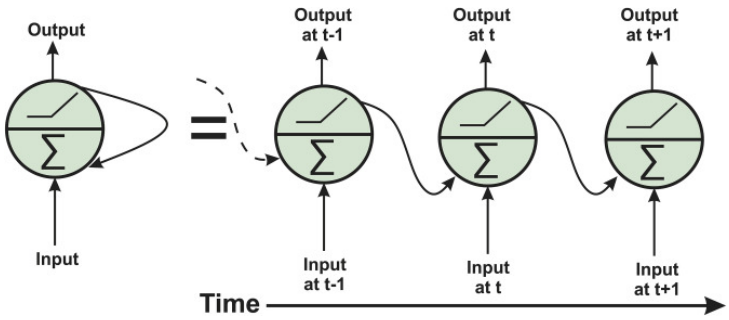


Figure 2.4: Comparison of rolled RNN (left) and unrolled RNN (right) [46]

dependencies in sequential data. While the repeating module in RNNs have a very simple structure, LSTMs consist of four interacting components. Figure 2.5 illustrates the internal design of a LSTM.

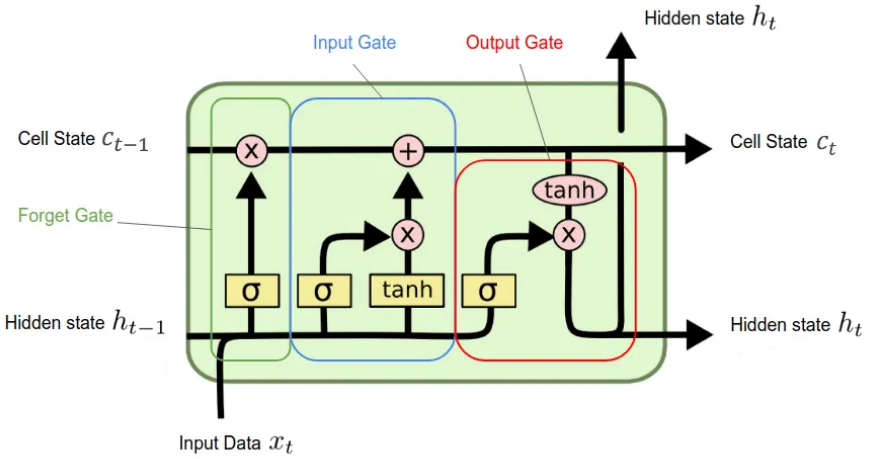


Figure 2.5: Structure of a LSTM module adapted from [48]

The key element of a LSTM is the LSTM cell c_t which may be considered as memory unit where information can be read, written or deleted. Access to the cell is controlled through an input, output and forget gate which determine on whether new input is fetched, irrelevant information should be erased or the hidden state h_t at time step t should be affected by the cell's content. More technically, the LSTM implements pointwise multiplication, addition, sigmoid activation and TanH activation functions to model the functionality of these gates, thus allowing the LSTM module to learn the temporal relation between sequential input tokens and the output respectively. Since it's invention, many variants of the LSTM have been proposed and further refined for different application domains such as the Convolutional LSTM (ConvLSTM) [49] which is used for spatio-temporal predictions for example.

2.3.3 Machine Learning Variants

Apart from gradient-based learning methods such as supervised learning 2.3.4, various other ML techniques exist. The following paragraphs provide a brief overview of the most common learning techniques used in ML.

Reinforcement Learning (RL) is one of the three fundamental ML paradigms next to supervised learning and unsupervised learning. In RL, an agent is trained on how to interact with its environment based on the notion of a cumulative reward. Typically, an environment is stated as Markov Decision Process (MDP) where a transition from one state to another state given a specific action is associated to a reward. The main difference between RL and supervised learning is that the correct solution does not have to be provided to the algorithm (no labelling of the input data is required). The aim of a RL agent is to find the optimal balance between exploration of new territory and exploitation of its current knowledge by learning a policy π of state and action pairs maximizing the expected cumulative reward to achieve an optimal solution.

Imitation Learning (IL) is a ML concept similar to RL but instead of providing a scalar reward as feedback for transitioning from one state to another state in the environment given a specific action, the agent receives a direct demonstration (an action) of the task as feedback. These actions are gathered from a trainer, denoted as expert which the model learns to imitate.

The idea of Transfer Learning (TL) is to use an already trained NN for a new problem. Based on the large amounts of data required for training DNNs from scratch, TL provides a convenient possibility of slightly adapting the pre-trained model and apply it to a similar use case without the need to retrain all layers of the model. In a computer vision context for example, the classifier of an image recognition model may be redesigned and/or retrained while the backbone of the network (involves the entire down-sampling path) which is used for feature extraction is left unchanged. For this purpose, aside from the vastly reduced training time, much less training data is necessary as only the classifier needs to be fine-tuned. It shall be noted that the fine-tuned model should be applied on inputs which are similar to those the pre-trained model has been trained with, otherwise the accuracy of predictions might still be low.

Finally, unsupervised learning is a ML method where unknown patterns and correlations in a dataset are learned without the need to label the data. Thus, an unsupervised learning algorithm analyzes the given data to discover potential similarities and differences in order to group the data into a previously undefined number of categories. Usually, unsupervised learning algorithms are utilized for clustering, association and dimensionality reduction. Clustering is the process of grouping data points based on similarity, difference, structure and/or pattern. Examples of clustering algorithms are K-Means, DB-Scan and hierarchical cluster analysis algorithms. Association is the analysis of finding correlations between the data points where the apriori algorithms may be noted as an example. Last but not least, dimensionality reduction aims at finding the least amount of variables to express the given data such that the algorithm does not overfit on the training data

and may also be used on other datasets. One example of a dimensionality reduction algorithm is Principal Component Analysis (PCA).

2.3.4 Supervised Learning

Supervised learning is a ML paradigm and one of the most commonly applied techniques to solve a specific problem using ML. The basic idea of supervised learning is based on the knowledge of the correct solution to a given problem. Supervised learning approaches may be split into two categories: classification and regression. In classification, an algorithm is designed to assign the correct category to each desired data instance. For example in semantic segmentation 2.4.1, each pixel of an input image is assigned a class label. Regression deals with the problem of finding a relationship between dependent and independent variables while the goal is to project their further relation. For instance, based on the stock price of the last few weeks, the stock price of tomorrow shall be projected. Apart from artificial NNs, several other types of supervised learning techniques exist such as Naive Bayes, linear and logistic regression, Support Vector Machine (SVM), k-nearest neighbour and random forest. This section only focuses on the explanation of supervised learning using artificial NNs.

An artificial NN, also denoted as model or network is trained on a dataset which consists of individual input and label pairs where the input is a data item the model is intended to be trained on and the label provides the actual solution to the input. The major drawback of supervised learning algorithms is that the input data must be labeled prior to training the model. Generally, the learning process of an artificial NN can be split into training phase and validation phase which are iteratively executed. Each iteration of the learning process is also referred to as epoch. For every epoch, the model is trained on a large amount of training data by performing forward propagation where an error term, denoted as cost or loss between the provided solution and the predicted solution is computed. Afterwards, the backpropagation algorithm is applied where the parameters of the model are recursively adjusted based on the partial derivatives of the loss function of these parameters. Once the training phase of the network for a given epoch is completed, the model is evaluated on a different portion of the dataset denoted as validation data. In the validation phase, the performance of the artificial NN is tested with respect to an evaluation metric determining the accuracy of the model. The evaluation metric should be interpretable and have a task-specific or domain-specific meaning as well as a correlation between the loss function and the evaluation metric is preferable, as a decrease in the loss might not lead to an increase in accuracy otherwise. Based on the change in accuracy over various epochs, the learning rate which is a fundamental hyper-parameter in gradient-based learning methods such as supervised learning, may be increased or decreased resulting in a more efficient training of the network.

In order to train the network appropriately, the loss function has to be sufficiently minimized. For this purpose, an optimization algorithm also denoted as optimizer is required which searches the parameter space of the model's weights and biases and adapts the learning rate to reduce the overall loss and converge to an optimal minimum. Various

optimization algorithms exist such as Gradient Descent, Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent, SGD with Momentum, Nesterov Accelerated Gradient (NAG), Adaptive Gradient Descent (AdaGrad), Root Mean Square Propagation (RMS-Prop), AdaDelta and Adaptive Moment Estimation (Adam). These optimization algorithms may be roughly categorized into two sets depending on the usage of a constant or dynamic learning rate. Optimization algorithms relying on a constant, manually selected learning rate have the disadvantage that the choice of learning rate heavily impacts the convergence speed. Moreover, variants of the Gradient Descent algorithm tend to converge to local minima. While optimization algorithms building on a dynamic learning rate allow to select a different learning rate for each parameter, those algorithms are more complex and quite computationally expensive. The following paragraph provides an introduction to the most basic, but also the most commonly used optimization algorithm: Gradient Descent.

The Gradient Descent algorithm is an iterative first-order optimization algorithm meaning that the first-order derivative of the loss function is calculated. Therefore, not any given function may be used for the computation of the loss as the loss function must be differentiable. Furthermore, the cost function is ideally convex as the algorithm may also converge towards local minima. For convex functions, the local minima also corresponds to its global minima. The parameters are updated by subtracting the scaled gradient of the current value from the current value. Mathematically, the Gradient Descent algorithm is formulated as follows: $\theta_{n+1} = \theta_n - \eta \nabla f(\theta_n)$ where θ_{n+1} is the parameter's updated value, θ_n is the parameter's current value, η is the step size, also denoted as learning rate and $\nabla f(\theta_n)$ is the gradient of the parameter's current value. As already mentioned, the learning rate is an essential parameter of the Gradient Descent algorithm. If the learning rate is too low, convergence may take forever. If the learning rate is too high, the algorithm might not converge or even diverge. Finally, the parameters are updated only once for every epoch as the entire dataset is used for computation. Thus, the Gradient Descent algorithm might have a large memory footprint as well as requires a large number of epochs to converge.

In order to partially overcome this problem, the SGD algorithm updates the parameters of the model after each iteration by only considering one randomly fetched training sample vastly reducing time for convergence. As the training samples are randomly selected, the parameters have high variance and fluctuations in the loss might occur. The Mini-Batch Gradient Descent algorithm tries to find a balance between Gradient Descent and SGD by updating the parameters of the model after a batch of training samples has been processed.

Finally, an advanced optimization algorithm using dynamic learning rates and second-order derivatives is addressed. Adam [50] is a first-order and second-order momentum based optimization algorithm of stochastic objective functions combining the benefits of AdaGrad and RMS-Prop. Adam uses an individual learning rate for each parameter of the model while the parameters are adapted with respect to the exponentially decaying average of the gradient and the exponentially decaying average of the squared gradient.

Thus, Adam uses the mean and uncentered variance of the gradients to rapidly improve convergence.

Specifically, when a model is trained, a set of data samples such as images denoted as a batch is passed on to the model. Each image is received at the input layer of the artificial NN and sequentially processed by all layers of the network to compute the output of the model. This process is called forward propagation or forward pass. Suppose, we consider the fully-connected NN shown in figure 2.6 with only one hidden layer consisting of two neurons and a single-neuron output layer.

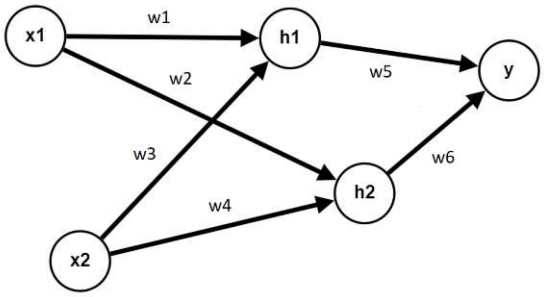


Figure 2.6: Structure of a simple FFNN

During forward propagation, the intermediate variables as well as the activations of the neurons are computed based on the model's current parameters. For instance, the intermediate variables and activations of the neurons in the hidden layer and in the output layer for the network displayed in figure 2.6 are calculated according to equations 2.1 - 2.6.

$$h_1 = x_1 * w_1 + x_2 * w_3 + b_1 \tag{2.1}$$
$$h_2 = x_1 * w_2 + x_2 * w_4 + b_2 \tag{2.2}$$
$$h_{1a} = \phi_1(h_1) \tag{2.3}$$
$$h_{2a} = \phi_2(h_2) \tag{2.4}$$
$$y = h_{1a} * w_5 + h_{2a} * w_6 + b_3 \tag{2.5}$$
$$y_a = \phi_3(y) \tag{2.6}$$

where $\phi_i(x), i \in [1..3]$, are neuron activation functions. Without the loss of generality, the computation of the intermediate variables and the hidden activation vectors based on a network with only one hidden layer for an input $x \in \mathbb{R}^n$ where n is the dimensionality of x may also be stated according to equations 2.7 - 2.10.

$$h = W_1 * x + b_1, W_1 \in \mathbb{R}^{m \times n}, b_1 \in \mathbb{R}^n \tag{2.7}$$
$$h_a = \phi(h), h \in \mathbb{R}^m \tag{2.8}$$
$$y = W_2 * h_a + b_2, W_2 \in \mathbb{R}^{o \times m}, b_2 \in \mathbb{R}^m \tag{2.9}$$
$$y_a = \hat{\phi}(y) \tag{2.10}$$

While the computational graph of the network is traversed in forward direction during forward propagation, the backpropagation algorithm traverses the network in reverse order and updates the parameters of each layer's neurons based on their relative impact on the loss by applying the chain rule consecutively. The backpropagation algorithm is the core for training NNs. Due to the nature of the chain rule, the computation of the gradient of each neuron's parameters only depends on the neuron's hidden state which was computed by forward propagation and on the gradient of each neuron's parameters that are on the path between the current neuron and the respective output neuron. For this reason, the backpropagation algorithm allows for high-parallelism in the overall computation of each neuron's loss making it the state-of-the-art training algorithm for NNs as it can be efficiently utilized to run on multiple Graphical Processing Unit (GPU)s simultaneously.

From a more theoretical perspective, the chain rule formally states that the derivatives of two differentiable functions represent the derivative of the composition of these two functions. Suppose we have two functions $z = f(y)$ and $y = g(x)$. Then, the chain rule given by equation 2.11 allows to express the partial derivative of the dependent variable z with respect to the independent variable x given the intermediate dependent variable y.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.11}$$

The objective of the backpropagation algorithm is to compute the gradients of the cost function C with respect to the model parameters. Based on the artificial NN given in figure 2.6 and by disregarding the bias parameters for the sake of simplicity, the partial derivatives $\frac{\partial C}{\partial W_{1,2}}$ need to be calculated according to equations 2.12 - 2.17. Therefore, the gradient of C with respect to the output layer y_a is computed first. Furthermore, the gradient of C with respect to the intermediate variable y is calculated while an elementwise multiplication to account for the activation function is necessary. In the next step, the gradients of the model parameters used for the output layer computation are obtained according to equation 2.14. Based on equation 2.14, we need to further traverse the network in backward direction in order to calculate the gradient of $\frac{\partial C}{\partial W_1}$. The gradient of C with respect to the hidden layer output h_a is obtained in equation 2.15. For computing the gradient of C with respect to the intermediate variable h, the activation function has to be considered again. Finally, the gradient of C with respect to the model parameters used for the hidden layer computation are computed according to equation 2.17.

$$\frac{\partial C}{\partial y_a} \in \mathbb{R}^o \tag{2.12}$$

$$\frac{\partial C}{\partial y} = \frac{\partial C}{\partial y_a}\frac{\partial y_a}{\partial y} = \frac{\partial C}{\partial y_a} * \hat{\phi}(y) \in \mathbb{R}^o \tag{2.13}$$

$$\frac{\partial C}{\partial W_2} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial W_2} = \frac{\partial C}{\partial y}h_a^T \in \mathbb{R}^{o\times m} \tag{2.14}$$

$$\frac{\partial C}{\partial h_a} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial h_a} = W_2^T\frac{\partial C}{\partial y} \in \mathbb{R}^m \tag{2.15}$$

$$\frac{\partial C}{\partial h} = \frac{\partial C}{\partial h_a}\frac{\partial h_a}{\partial h} = \frac{\partial C}{\partial h_a} * \phi(h) \in \mathbb{R}^m \tag{2.16}$$

$$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial h}\frac{\partial h}{\partial W_1} = \frac{\partial C}{\partial h}x^T \in \mathbb{R}^{m\times n} \tag{2.17}$$

Thus, given the model's initial parameter initialization, when a model is trained, forward propagation and backward propagation are alternated as they essentially depend on each other.

Fitting the model on a dataset is a non-trivial task, as the model must be represented with descent complexity to capture the underlying distribution in the dataset. Usually, the distribution in the dataset is either assumed to be Gaussian or non-Gaussian. When sampling from data of a given distribution, we aim at either standardizing the data by having a mean of zero and a standard deviation of one in the data for Gaussian distributions or normalizing the data to a common range e.g. between zero and one for non-Gaussian distributed data of various scales. Many ML algorithms benefit from this pre-processing step as bias and variance play a vital role in finding a statistical fit for an unknown target function. The complexity of the model is the second fundamental parameter. Figure 2.7 provides an argument on the importance of the model's complexity. Underfitting is a phenomenon that occurs if the model has a high bias and is too simple to represent the desired target function while overfitting refers to an overestimation of the target function where the model has high variance and also attempts to capture outliers/noise in the data. Commonly, the problem of underfitting can be identified by the model's poor performance on the training and test data and tackled by either increasing model complexity or training data size. On the other hand, the problem of overfitting is much harder to deal with, thus it is discussed in the following in more detail.

In order for DNNs to generalize well on new data, overfitting of the model to the training data must be prevented. Generally, the term overfitting describes the problem that the model achieves a low loss and high accuracy on the training data as the model tries to fit to the noise and variance in the training data while a much lower or even poor prediction confidence on new data is given. In order to avoid this issue, regularization techniques have to be implemented in the design of the model and/or the learning process. The most common regularization techniques in deep learning are L1 & L2 regularization,

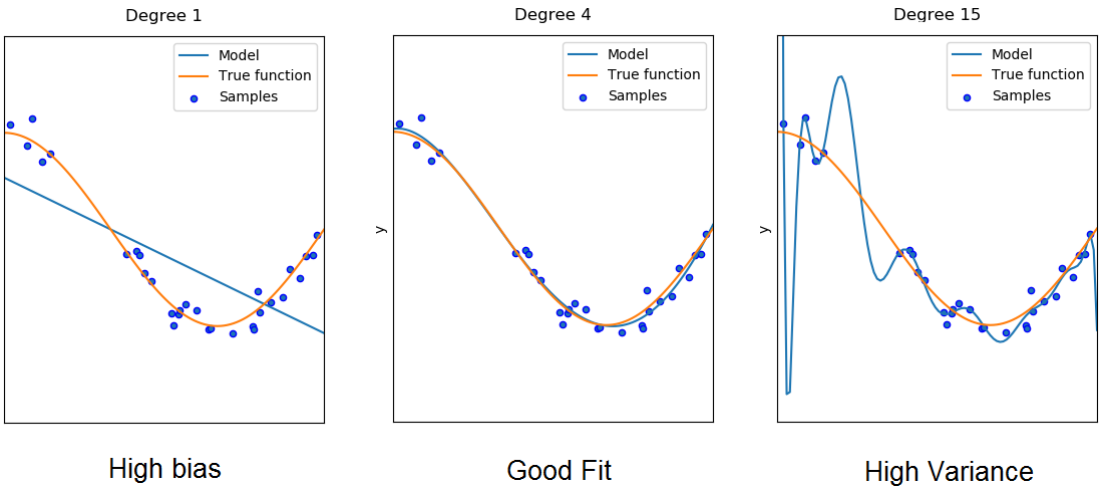


Figure 2.7: Bias-variance trade-off for optimal model generalization from [51]

dropout, batch normalization, data augmentation and early termination. This section will conclude with a brief explanation of each of the stated methods.

The basic idea behind L1 & L2 regularization is based on the assumption that smaller parameter values result in simpler and more generalized models. Hence, the respective loss function is extended by a regularization term penalizing large parameter values. This ensures that the overall magnitude for the model's parameter values decreases. Depending on the implementation of L1 or L2 regularization, the regularization term varies in how the model's parameters are influenced. For L1 regularization, the sum of absolute values of the parameters is penalized. Mathematically, the L1 regularization term extended loss function is given by $loss = loss + \lambda * \Sigma_{i=1}^{N} |w_i|$. Contrarily, the L2 regularization term penalizes the sum of squared parameter values and acts as weight decay to drive the parameters to decay towards zero. The L2 regularization term extended loss function can be formulated as follows: $loss = loss + \lambda * \Sigma_{i=1}^{N} |w_i|^2$. λ is a hyper-parameter in the range between 0 and 1 specifying how much the regularization term contributes to the overall loss. L1 regularization is robust to outliers in the data, as parameters may actually be zero and thus, discard some features entirely while for L2 regularization, parameters may only take non-zero values, preserving a slight impact of each feature.

Dropout is a regularization technique directly implemented in the design of the network. Based on a selected probability, inputs of a hidden unit are randomly set to 0, discarding their influence in all further processing steps. Specifically, neurons of two consecutive hidden layers are randomly disconnected. This procedure is only applied during the training phase of a model.

The variations in mean and standard deviation of the samples in each batch has a large impact when training the model where a batch mean of 0 and a batch standard deviation of 1 is more optimal. Thus, batch normalization overcomes this issue by computing the batch mean and batch standard deviation in order to normalize the samples in each

batch. During the validation phase, the already precomputed batch mean and batch standard deviation are used, leading to a faster learning convergence of the network.

Especially for small datasets, models tend to overfit. Therefore, data augmentation proves to be among the most efficient regularization techniques to battle this challenge. Data augmentation defines an artificial extension of the training dataset where e.g. a set of images from the training dataset is randomly transformed and appended to the training dataset. With respect to computer vision applications, transformations for images may include color jittering (brightness, contrast, saturation and hue), flipping, rotation or cropping. The specifically applied transformations have to be selected based on the input space of the domain, e.g. in autonomous driving, no vertically flipped camera images will be recorded. Thus, there is no point of making the model robust against this type of inputs. Additionally, no labeling is required, as the already labeled training data is modified. If data acquisition is a tedious and cumbersome task for the respective ML problem, data augmentation is essential.

Finally, early termination considers the problem of overfitting by monitoring the trend of the validation loss during the learning process. Ideally, as the training loss declines, the validation loss should also monotonically decrease and follow the trend of the training loss. At a given epoch, the validation loss might not decline but increase, indicating that further training might lead to an overfitted model. Thus, the learning process is terminated earlier.

Generally, a combination of a subset of the described regularization methods may even further improve the model's ability to generalize well while a mix of all regularization techniques may even result in a decline in the model's accuracy. In summary, a trade-off between bias and variance has to be found to arrive at a statistical fit which provides an optimal generalization of the model on new data.

2.4 Computer Vision

The science of computer vision is a field of AI aiming at analyzing digital 2D/3D camera images or videos in order to comprehend the visual world and extract the content of a given scene. The overall goal of computer vision is to replicate human vision and enable machines to have a high-level understanding of what is depicted in a scene given specific visual inputs. Machines may use this information to determine on how to optimally interact with the environment based on what they identified in the visual perceptions. Especially for cyber-physical systems such as robots or autonomous cars, computer vision is of major interest as those systems have to carefully perceive their environment in order to make well-founded decisions on their next move. Specifically, autonomous driving agents need to satisfy strong safety and real-time requirements regarding perception, planning and control to safely and reliably perform a specific task where computer vision algorithms are crucial for perception. Despite the ranging capability and vast applicability of different sensor systems such as Radio Detection and Ranging (Radar) or 2D/3D Light Detection and Ranging (LiDaR) sensors for environmental sensing, camera systems

have the advantage of allowing images to be further analyzed by advanced computer vision algorithms building upon AI. Furthermore, the transportation infrastructure is designed to be visually interpreted based on how humans drive e.g. traffic lights, traffic signs or lane boarders may only be detected using a visual camera system. Contrarily, camera systems have the drawback that computer vision algorithms are sensitive to illumination. Hence, as safety can be concerned as the most important requirement in autonomous driving, especially for harsh weather conditions, sensor fusion will be required to compensate for the disadvantages of multiple different sensor systems nevertheless.

While the smallest unit of a digital image is related to as pixel, computer vision is more complex than the sum of interpreting each pixel in the image individually. One of the core components of computer vision algorithms is feature extraction. Features are specific characteristics of individual parts of an image. Depending on the computer vision algorithm, these features are either hand-crafted by a human or learned using ML. The computer vision algorithm further uses these features to reason about the content of the image. In case of state-of-the-art DNNs for image processing which build upon the fundamentals of CNNs, multiple convolutional layers are stacked to perform feature extraction. For each convolution operation, the network learns the weights of the applied kernel filter matrix. These feature extraction networks also referred to as backbones are used to extract a wide range of features, from low-level features such as distinct pixel colors, lines or edges in the upper hidden layers to high-level features such as distinct object shapes in the lower hidden layers. Important attributes of strong features are repeatability, robustness and uniqueness. In order for the model to generalize well, the extracted features must be representative of the described object category in general (repeatability), allow to characterize an object despite the noise in the image (robustness) as well as distinguish an object from other objects (uniqueness). Based on the structure of DNNs for image processing, features of the upper hidden layers are simpler than features of lower hidden layers as the receptive field of the artificial neurons in the upper hidden layers is much smaller in comparison to the receptive field of the artificial neurons in the lower hidden layers. Thus, these neurons only have a local view of the entire image and less information available. In general, the receptive field of an artificial neuron defines the proportional region of the input space (patch) that a stimuli is capable of triggering that neuron. Rephrased, the receptive field of an artificial neuron is defined by the area of the input space that produces an output feature. The size of the receptive field is especially important for neurons in the last convolutional hidden layers, as they must be able to consider all essential information that is necessary for further processing. For example in image classification, the size of the receptive field of the neurons in the last convolutional hidden layer in a CNN must be as large as the largest object to classify, otherwise classification results may decrease as the object is never fully viewed. The concept of a neuron's receptive field only applies to local operations such as convolution or pooling as each neuron in a fully-connected layer has access to all information available.

Apart from autonomous driving, computer vision algorithms are also used for face recognition in social media, brain tumor diagnosis in healthcare using biomedical image segmentation, customer and item detection in retail stores, defect detection in manufacturing as well as in military and space technology. Due to its vast application possibilities, there are many tasks computer vision algorithms can be successfully applied to. The main tasks of computer vision may roughly be categorized into image classification, image segmentation, object localisation, object detection and object tracking. Figure 2.8 illustrates the difference between the individual computer vision tasks.

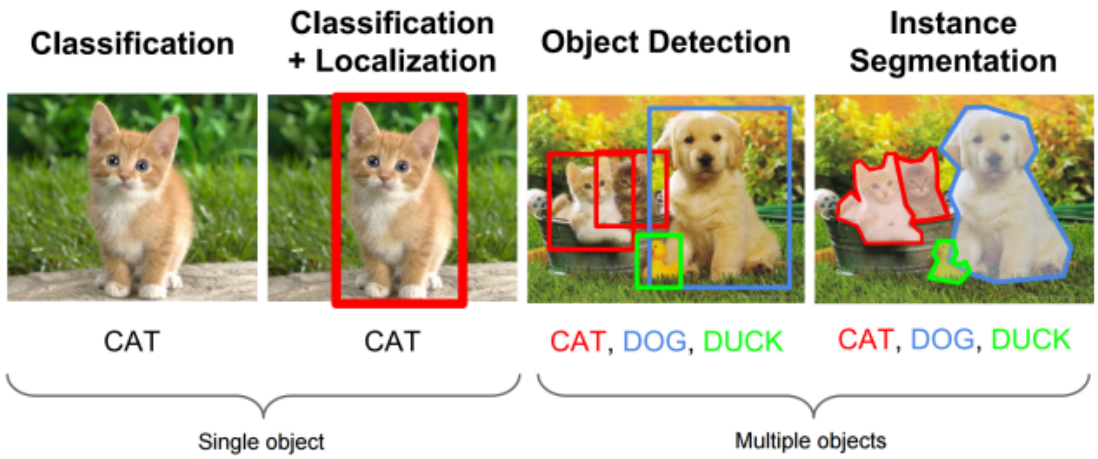


Figure 2.8: Visualization of the various computer vision applications from [52]

In image classification, predefined class labels are assigned to entire input images. The class labels are represented as integers ranging from zero to $\#classes - 1$ where each number corresponds to a user-specific class. An artificial NN e.g. a CNN is trained on a set of sample images that have already been labelled and expected to classify new images accordingly.

Unlike image classification, image segmentation describes the task of assigning a predefined class label to each pixel of the input image rather than the entire input image. Therefore, instead of a single scalar integer class label, a 2D or 3D segmentation map with the same dimensionality than the input image must be provided for each input sample for training. In image segmentation, the input space is divided into subregions allowing the artificial NN to differentiate between various objects as well as objects and the background. For further processing, each identified object can be extracted from the image using its corresponding pixel mask from the segmentation map. Depending on if a pixel mask for each individual object or the object category in general is sufficient, we can distinguish between semantic segmentation and instance segmentation.

Object localisation tackles the problem of assigning a 2D or 3D bounding box to identify an object in an image. Once the object has been located, it may also be classified. Depending on the format of the label, the coordinates of the bounding boxes are either

provided in terms of the center coordinates of the bounding boxes and the width, height and depth of the bounding boxes or by directly providing the corner coordinates of the bounding boxes.

The primary objective of object detection combines object localisation and object classification by first assigning a 2D or 3D bounding box to identified objects in an image which are then further classified according to predefined class labels. In contrast to object localisation, the number of objects in a given image sample is unknown. Thus, the output space of object detection algorithms varies in length as the bounding box coordinates as well as the class label are required for each detected object.

Finally, object tracking is defined as the problem of detecting multiple objects in images and uniquely correlating these objects across multiple images. In comparison to object detection, the output vector is further extended by a target id which describes a particular instance of a detected object category. By incorporating an additional target id label, each instance of an object may be tracked individually. This allows to predict the trajectories of objects based on the same detected objects in previous images.

There are many more specific computer vision tasks such as image restoration, image stitching, scene reconstruction, pose estimation or motion analysis. For the purpose of this thesis, we are going to provide a more detailed perspective on the tasks of semantic segmentation and object detection as those are relevant for the work incorporated in this thesis.

2.4.1 Semantic Segmentation in Computer Vision

Semantic segmentation is a type of image segmentation and depicts the task of separating object categories from each other as well as the background in an image. More specifically, a predefined class label is assigned to each pixel of an input image which is also referred to as dense prediction. For this purpose, an artificial NN is provided with a set of image samples and a corresponding segmentation label called the segmentation map for training. The segmentation map has the same dimensionality as the input image sample while each pixel of the segmentation map is assigned the dedicated integer class label of the object category it belongs to. For instance, for each RGB input image with $height \times width \times 3$, a segmentation map with $height \times width \times 1$ is provided as target. Alternatively, the segmentation map may also be one-hot encoded meaning that there exists an individual channel for each class in the segmentation map where only the pixel mask of the segmented object class is present while all other pixels are zero. In this case, the segmentation map has dimensionality $height \times width \times C$ where C is the number of object categories. In order to arrive at a segmentation map with only one channel as before, the channels can be merged by using the maximum argument of each depth-wise pixel vector. One-hot encoding has the advantage that categorical data can be represented in an efficient way for the ML algorithm to learn.

When training a NN for semantic segmentation, the categorical cross-entropy loss function proves to be quite promising. Categorical cross-entropy loss is a loss function that allows

to deal with categorical data and expects the labels to be one-hot encoded. DNNs used for semantic segmentation tasks that only require to separate an object or some other specific region from the background may also use the binary cross-entropy loss function. Binary cross-entropy loss does not require the labels to be one-hot encoded and returns a scalar floating point value between zero and one for each pixel of the segmentation map where a value closer to zero indicates that the respective pixel belongs to the background and vice versa. For validation, alongside pixel accuracy and Intersection over Union (IoU) also denoted as Jaccard coefficient, the Sørensen–Dice coefficient also denoted as Dice Similarity Coefficient (DSC) provides a precise and strong metric for evaluating the performance of a semantic segmentation model. Each of these metrics is bound to the range from zero to one where zero indicates no correspondence and one refers to perfect overlap. Pixel accuracy measures the proportion of the correctly classified pixels in the segmentation map. While this metric seems to be sufficient for validating the performance of a semantic segmentation model intuitively, it yields a poor performance when dealing with class imbalanced data. In most real world datasets, class imbalance is present meaning that one or more classes dominate the input space e.g. an image. In order to tackle this problem, we have to consider more than only the true positives. For this purpose, IoU measures the area of overlap divided by the area of union between the predicted segmentation map and the ground truth label and is given by $\frac{TP}{TP+FP+FN}$ where TP refers to the true positives, FP refers to the false positives and FN refers to the false negatives when comparing the model's prediction with the ground truth labels. Contrarily, DSC describes two times the area of overlap divided by the number of pixels in both images between the predicted segmentation map and the ground truth label and is given by $\frac{2\times TP}{2\times TP+FP+FN}$. While IoU and DSC have a positive correlation, the main difference between IoU and DSC is illustrated when computing the average score over a set of predictions. Generally, IoU penalizes instances of wrong classifications (FP and FN) harder. Therefore, IoU is more sensitive to classification outliers and tends to converge towards worst-case performance while DSC tends to stick closer to average performance of the model. Apart from being used as validation metrics, IoU and DSC may also be formulated as differentiable functions and used as loss functions by negating the coefficient's value. Hence, a high metric score is directly related to a low loss when an optimizer aims to minimize the loss function respectively.

However, the architectural design of the NN used for learning a given semantic segmentation problem is fundamental. To this end, a basic approach is to use a CNN and preserve the full resolution of the input dimensions at each layer of the network by using a respective padding. Thus, the network directly learns a set of transformations of feature mappings between the input images and their corresponding segmentation maps. Obviously, such an algorithm is quite computationally expensive. In order to reduce the computational costs and maintain expressiveness, an encoder-decoder structure proves to achieve promising results. The encoder also denoted as downsampling path in this context reduces the spatial resolution of the images while increasing the number of feature maps (channels). Similar to a CNN, the encoder extracts features from the input images which are further processed and scaled-up by the decoder also denoted as upsampling path to the

full resolution of the input images. The encoder uses convolution and pooling operations to manipulate the input data and reduce the spatial resolution. On the other hand, several different methods for upsampling exist. The most popular approaches for upsampling are nearest neighbour interpolation, bilinear interpolation and transpose convolution. While nearest neighbour interpolation and bilinear interpolation are mathematical operations involving no tunable parameters, transpose convolutions allow to learn the parameters of the upsampling operation. In comparison to a regular convolution which computes the pixel value by calculating the dot product of an image patch with its kernel matrix, an element-wise multiplication of a single feature value with the kernel matrix of the transposed convolution is performed to produce an upsampled feature map. Despite the advantage of requiring less computational resources and learnable encoder/decoder operations, the network suffers from the lack of local information in the upsampling path. Thus, the ML algorithm is only capable of computing coarse-grained segmentation maps. In order to face this limitation, the decoder is split into multiple stages similar to the encoder structure where additional skip connections between the encoder and decoder stages are added. At each stage of the decoder, the lower spatial resolution feature maps are upsampled and combined with the equivalent spatial resolution feature maps of the respective encoder stage. The combination of the feature maps from the encoder and the upsampling path may either be implemented in terms of an addition or a concatenation. Ronneberger et al. proposed the U-Net architecture [53] which follows a symmetric encoder-decoder architectural design incorporating skip connections between the encoder/decoder stages. Figure 2.9 depicts the standard U-Net architecture based on the details of the reference paper.

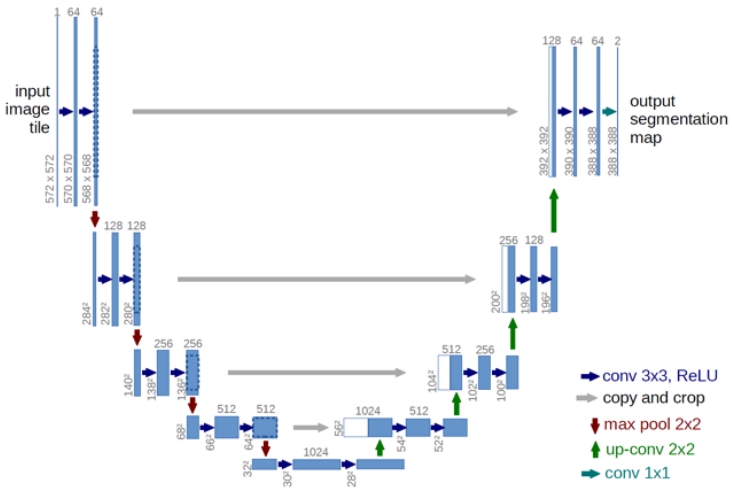


Figure 2.9: Schematic of the original U-Net architecture [53]

According to the original paper, the U-Net architecture consists of a contracting path (encoder) and an expansive path (decoder). Each block of the encoder is composed of two

3x3 convolutions where each convolution is followed by a ReLU activation. Afterwards, the feature maps are downsampled and the feature channels are doubled using a 2x2 max pooling operation with a stride size of two. For each block of the decoder, the feature maps are upsampled, a 2x2 convolution is applied to reduce the number of feature channels by a factor of two and the upsampled feature maps are concatenated with the corresponding feature maps from the encoder. Finally, two 3x3 convolutions followed by a ReLU activation are performed. For the last layer of the expansive path, a 1x1 convolution is necessary to map each 64 dimensional feature vector to a feature vector with a channel depth corresponding to the number of classes.

U-Net was trained with very few annotated larger scale image samples and a batch size of one. Additionally, data augmentation in terms of random elastic deformations was applied to achieve the necessary robustness and invariance properties in order for U-Net to perform well in biomedical image segmentation use cases. For computing the loss, a pixel-wise soft-max function in combination with the cross-entropy loss function was utilized. In order to compensate the class imbalance in the training data set and to emphasize the boarders between the individual regions in the segmentation map, the weight map for each ground truth label used in the loss function was pre-computed.

While the standard U-Net architecture achieves remarkable results for various biomedical image segmentation applications, many more advanced extensions of U-Net have been proposed such as [54] [55]. Zhou et al. proposed U-Net++ [54] which redesigns the skip connections of the original U-Net architecture by implementing convolution layers as skip connections as well as by additionally utilizing nested, dense skip pathways. On the one hand, this allows U-Net++ to bridge the semantic gap between the feature maps of the encoder and decoder subnetworks. On the other hand, gradient flow is improved due to the dense skip connections. Last but not least, U-Net++ also relies on deep supervision during training. Oktay et al. proposed Attention U-Net [55] which adds an additional additive attention gate before each stage of the upsampling path. Based on the upsampled feature maps and the contextual information provided along the skip connection from the corresponding encoder block, the upsampled features are scaled with attention weights calculated by each attention gate of the decoder subnetwork.

2.4.2 Object Detection in Computer Vision

Object detection is one of the most researched areas in computer vision and a core component in many computer vision applications. In particular, object detection aims at localizing and classifying a previously unknown number of objects in a given scene image. Recent computer vision algorithms are deployed at multiple scales, from edge devices and embedded mobile computing platforms to large scale server clusters and clouds. State-of-the-art object detection algorithms must comply with strong application requirements while being limited by the computational resources of the integrated environment. Modern computer vision treats object detection as a regression problem where in addition to the class labels, a set of numbers related to the bounding box coordinates of each object in the image are predicted. Based on traditional computer vision algorithms, one of the first approaches was to use an external method to generate region proposals of where objects are assumed to be located in the image which are further sequentially passed to a CNN for classification. Specifically, this procedure known as R-CNN [38] proposed by Girshick et al. uses selective search to extract region proposals also denoted as RoI from an input image, resizes these RoI to a fixed shape and subsequently forwards those through a CNN to a SVM for classification and bounding box prediction. While this approach achieves a high object detection accuracy, it suffers from the multiple sequential RoI passes to the CNN resulting in a slow and computationally costly network. In order to avoid passing multiple RoI to the CNN one by one, Girshick et al. proposed Fast R-CNN [39]. Instead of externally computing and cropping the RoI of the input image, the entire image is passed to the CNN where the RoI are again computed on the input image using an external method and projected on the resulting feature map. Furthermore, the feature map is passed on to a Fully Connected Neural Network (FCNN) for classification and bounding box regression. The RoI projection on the output feature map of the CNN is implemented in terms of a RoI pooling layer which receives the computed RoI as well as the feature map from the CNN as input and performs a max pooling operation on its variable-shaped input and outputs a fixed-size feature map. As this algorithm is more efficient when compared to R-CNN, the reliance on an external RoI computation is undesired. For this purpose, Ren et al. proposed Faster R-CNN [40] that substituted the external RoI computation procedure of Fast R-CNN with a Region Proposal Network (RPN). The RPN generates region proposals based on the feature maps from the CNN which are both provided as input for the RoI pooling layer. Figure 2.10 illustrates the architecture of Faster R-CNN.

Faster R-CNN outperforms R-CNN and Fast R-CNN in terms of inference time (frame rate) as well as object detection accuracy. Despite its improvements and high object detection accuracy, Faster R-CNN is still too computationally expensive for an efficient deployment on embedded devices. Therefore, alternative object detection approaches have been proposed leading from two-stage detectors to one-stage detectors where the spatial information from the CNN is directly used for classification and bounding box regression instead of a region proposal network to avoid redundant computations. Hence, the object detection task can be performed in a single network. While two-stage methods

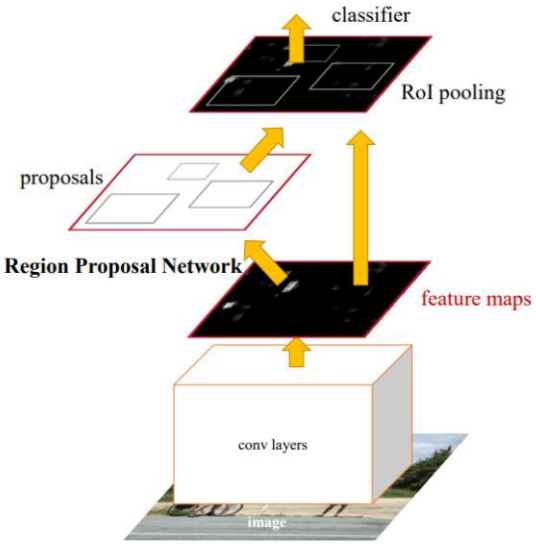


Figure 2.10: Representation of the Faster R-CNN architecture [40]

achieve a slightly higher object detection accuracy than one-stage methods, one-stage object detection algorithms are much faster. Liu et al. proposed SSD [37] which consists of a backbone model for feature extraction and a SSD head for classification and bounding box prediction. The FCNN is removed from the backbone model and the downsampling path preserving all of the spatial information of the input images at lower resolution is connected to the SSD head. The SSD head is composed of several convolution layers which successively decrease in size and where each convolution predicts objects at different scales. The spatial information of the activation maps of the last convolution layers of the SSD head are interpreted as the classification and bounding box predictions. The final prediction results are obtained by applying Non Maximum Suppression (NMS). The core of the SSD architecture relies on the receptive field which enables the prediction of objects at multiple scales. The structure of modern deep CNNs can be beneficially used for the backbone model as the input space is divided into a hierarchy of multiple local cells of varying size across the network's layers in terms of a pre-defined grid layout. Each cell is responsible for localizing objects in its specific region. Based on the size of the receptive field, lower layers are more capable of predicting small scale objects while higher layers have a larger receptive field enhancing the detection of large scale objects. Cells at the same layer have the same receptive field and the same amount, but different information available. Thus, the network is able to detect a given object independent of its position in the input space known as translation invariance. If no object can be located within a cell, the background class is assumed and no bounding box coordinates are predicted. For detecting multiple objects within a cell, various anchor boxes are assigned to each grid cell. Anchor boxes also denoted as prior boxes are defined in various sizes/scales and shapes/aspect ratios a priori. In the training phase, a set of anchor boxes is generated for every feature where the most fitting prior bounding box is matched with each object's

ground truth bounding box. Similar to semantic segmentation 2.4.1, IoU or DSC are commonly used to determine the default bounding box with the largest overlap indicated by a confidence value. The general objective function is composed of the weighted sum of a confidence loss and a regression loss where the confidence loss is a softmax loss over multiple class confidences and the regression loss is a Smooth L1 loss. The downside of SSD is the need for an extensive data augmentation strategy implying that a vast amount of training data is necessary for the model to perform well. Additionally, the detection accuracy for small objects might suffer from the poor feature generation in the lower layers of the network.

One of the most fundamental families of object detection architectures was introduced by Redmon et al. back in 2015 who proposed YOLO [28], the first real-time end-to-end object detector. Since then, many improvements have been made on the overall network architecture and the training process [32, 33, 34, 36, 29]. There is also an open source version of an improved YOLOv4 model called YOLOv5 [35] that lacks an official paper but was ported from C to Python to be more easily accessible to the deep learning community. While we use the state-of-the-art YOLOv7 object detector in this work, which is introduced in Section 4.2, the earlier iterations of the YOLO series illustrate many basic design concepts of the object detectors. For this purpose, the foundations of YOLO, namely YOLOv1 - YOLOv4 are discussed here.

Similar to SSD, YOLO basically divides the input image into a $S \times S$ grid where each cell is responsible for detecting objects in its region in terms of N bounding boxes, confidence scores and C classification labels. Figure 2.11 illustrates the object detection procedure performed by YOLO.

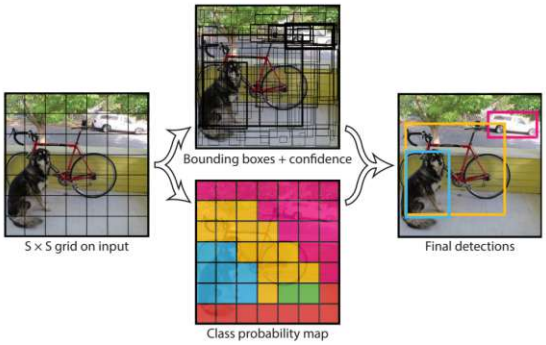


Figure 2.11: Illustration of the unified object detection approach applied by YOLO [28]

YOLO uses features from the entire image to predict all bounding boxes of all objects simultaneously. Each bounding box is represented by five predictions: x, y, width, height and confidence where x and y are the center coordinates of the bounding box relative to the grid cell coordinates while width and height are provided relative to the image dimensions. All four bounding box identifiers are normalized between zero and one with respect to the spatial dimensions of the input images. The confidence score is an indicator on how well the predicted bounding box of an object fits the ground truth bounding box

of this object and is usually computed using IoU. Independent of the number of predicted bounding boxes, C conditional class probabilities are predicted per grid cell where each conditional class probability is multiplied with the individual bounding box confidence score resulting in a class-specific confidence score for each bounding box. In order to arrive at the final object predictions, YOLO uses NMS to retrieve the final bounding boxes corresponding to the highest class-specific confidence score per object. To this end, YOLO encodes its predictions in a $S \times S \times (N * 5 + C)$ tensor.

Equivalent to SSD, YOLO does not predict the bounding box coordinates for each object in an image from scratch. Instead, various anchor boxes of different height, width, scale and aspect ratio are generated a priori at each grid intersection, also denoted as anchor points. Each grid cell is defined by the coordinates of an anchor point. The shape of the anchor boxes is based on the configuration of the anchor box sizes for each Feature Pyramid Network (FPN) head. These anchor box sizes are hyperparameters of the YOLO network. During training, the model outputs x, y, width and height corrections of the most promising anchor boxes in order to match a specific target bounding box. Due to the design of the model, these corrections may only be in a restricted range of adjacent grid cells, thus only a subset of anchor boxes, denoted as center prior anchor boxes may match a target bounding box. This procedure drastically simplifies the object detection problem and heavily reduces the computational cost.

While the architecture of the first YOLO implementation consisting of 24 convolution layers and two FCNN was rather simple and did not include multi-scale feature map predictions such as SSD, YOLOv4 is represented by a far more sophisticated one-stage architecture for instance which is depicted in figure 2.12.

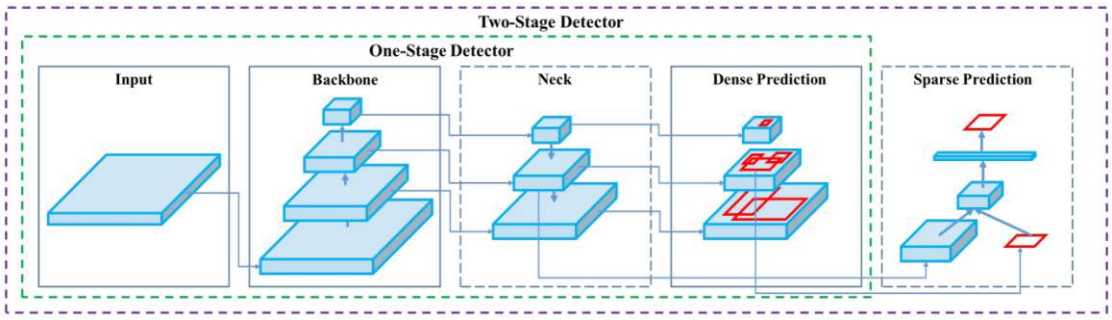


Figure 2.12: Illustration of the YOLOv4 architecture [34]

Generally, feature extraction is performed in the backbone of YOLOv4 where multi-scale feature maps of different backbone levels are extracted in the neck. Finally, the YOLO head is responsible for object classification and bounding box regression across multi-scale feature maps. In more detail, YOLOv4 uses Cross Stage Partial Network (CSPNet), specifically CSPDarknet53 [56] as a feature extraction backbone. Spatial Pyramid Pooling (SPP) [57] and Path Aggregation Network (PAN) [58] are integrated in the neck where SPP drastically increases the receptive field size and highlights context features and PAN performs parameter aggregation of different backbone levels

while an anchor based YOLOv3 head performs the dense predictions. Additionally, YOLOv4 applies the concepts of Bag-of-Freebies (BoF) and Bag-of-Specials (BoS) where either the training strategy is switched to improve object detection accuracy without increasing inference cost or object detection accuracy is significantly enhanced by slightly increasing inference cost. Specifically, YOLOv4 uses genetic algorithms to select optimal hyper-parameters, self-adversarial training, mosaic and mixup data augmentation, cross mini-batch normalization, dropout regularization, Complete Intersection over Union (CIoU)-loss for objectness computation, Distance Intersection over Union (DIoU)-NMS and many more advanced techniques.

Last but not least, DETR [4] is an alternative approach to object detection inspired by [8]. DETR addresses object detection as a direct set prediction problem and utilizes the self-attention mechanism of the transformer encoder-decoder network. The architecture is composed of a CNN backbone for feature extraction, a transformer encoder-decoder architecture and FFNN as prediction heads. Figure 2.13 depicts the DETR algorithm.

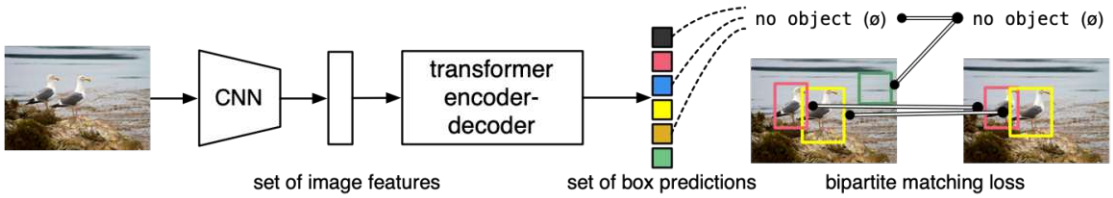


Figure 2.13: Visualisation of the DETR architecture [4]

A positional encoding is computed for the flattened set of image features extracted by the CNN backbone before those along with the positional encoding are passed to the encoder of the transformer. The decoder of the transformer is provided with a small set of learned positional embeddings, denoted as object queries as well as the output of the encoder to reason about the global image context and object relations. Each output embedding of the decoder is passed on to a shared FFNN which acts as the prediction head. The FFNN computes the final set of bounding box predictions along with the predicted class indices. During training, the network applies a set-based global loss in order to force unique predictions via bipartite matching. The strength of the DETR architecture relies in its anchor-free design as well as in the combination of a CNN backbone for feature extraction with the power of the transformer network to contextualize the global information of the processed image efficiently. Thus, DETR is able to accomplish state-of-the-art results compared to other object detection approaches.

CHAPTER 3

Scientific Methodology

The research method applied in this thesis follows a design and creation research strategy and builds upon academic literature as well as relevant specifications [59]. In computing research, the design and creation strategy focuses on developing new IT products called artefacts [60]. Such artefacts include constructs, models, methods and instantiations [61], whereby in most research, a combination of several artefacts contribute in creating new knowledge. As artefacts often represent computer-based products, the design and creation research strategy emphasises on analysis, explanation, argument, justification, and critical evaluation of the results to distinguish itself from product development.

In the context of design and creation research, the focus lies either on the artefact itself, (e.g. the IT application incorporates a new theory), the artefact as a vehicle to create new knowledge (e.g. the IT application in use) or on the process to create an artefact to create knowledge [62]. The focus of this thesis is two-fold and lies in the creation process as well as on how the artefact performs in its application domain. Specifically, this thesis creates knowledge in designing a computer vision-based object detection system applied in an autonomous racing setting capable of utilizing predicted human-attention for enhancing object detection robustness as well as provides insights on design challenges related to this novel approach.

Design and creation research is a problem-solving approach and builds upon the principles of system development [63]. The process typically involves five steps—awareness, suggestion, development, evaluation and conclusion. Whereby these steps are not rigid in order but instead form an interative cycle. Based on this iterative process, research in this work is conducted.

In particular, a comprehensive research methodology is employed, where we initially perform an in-depth literature research on the state-of-the-art of computer vision research, especially focusing on the topic of object detection. This encompassing research establishes the theoretical basis of this thesis as well as provides the baselines for the evaluation

of the Attentional Neural Network (AttNN) based dynamic object detection system. The primary challenge of this thesis relies on the design of the human-attention model (AttNN), the integration with the object detection model and how our system performs in its application domain. Assuming that humans persistently attend to RoI in their FoV, we apply the supervised learning paradigm to train the AttNN on the human-attention based RoI for given visual inputs. In order to create a dataset for training the human-attention model, we monitor the attention of the subjects on given track layouts while the manually driven F1Tenth racecar is in motion. To this end, we create a technical setup - also denoted as data-collection pipeline - composed of several hardware and software components including eye-tracking glasses which allows us to simulate the navigation of the F1Tenth racecar from a first-person perspective. This simulation is performed with a variety of subjects on multiple different indoor track settings such that the human-attention model is not biased on single drivers and generalizes well. Once the human-attention dataset has been created, we will explore several algorithms, learning strategies and human-attention label representations in order to assess which one fits best for modeling the AttNN and learning the relation between camera frames and associated human-attention regions.

For evaluating the overall system, the human-attention model is integrated with an object detection NN that is trained on a custom object detection dataset, denoted as F1Tenth object detection dataset. The human-attention dataset as well as the F1Tenth object detection dataset are gathered from the same track layouts. The experimentation phase will involve evaluating novel approaches for designing the AttNN based dynamic object detection system and comparing the performance of selected object detection baselines from state-of-the-art computer vision research trained on the F1Tenth object detection dataset with the AttNN based dynamic object detection system. We will quantify the impact of illumination perturbations on Mean Average Precision (mAP), among other relevant metrics. Furthermore, we demonstrate the object detection results of our approach compared with DETR along the predicted human-attention feature maps and the artificial attention on a given visual input. Finally, in order to evaluate the feasibility of our proposed approach for real-world autonomous driving/racing applications, we will deploy our system in the Robot Operating System (ROS) [64], validate the algorithm on a F1Tenth hardware car given custom indoor tracks and establish a collaboration of our system with Highly Automated Driving (HAD) agents. Concluding our research, the results of this comprehensive evaluation will be analyzed, interpreted and discussed.

3.1 Technologies

The following paragraphs give a short introduction about the most important tools and frameworks that have been used throughout this work. For this thesis, Python 3.9, PyTorch 1.11.0 and CUDA 11.6 have been used for developing, training and validating the AttNN (Section 4.1) and the object detection NN (Section 4.2) as well as for evaluating the AttNN based dynamic object detection system in simulation (Section 4.3.1) and the comparison to state-of-the-art computer vision research (Section 5). For the deployment

of the AttNN based dynamic object detection system (Section 4.3.2) on the physical F1Tenth hardware car [1], JetPack 5.0.2 including TensorRT 8.4.1, CUDA Deep Neural Network (cuDNN) 8.4.1, CUDA 11.4 and Python3.8 as well as ROS Noetic Ninjemys were installed on the NVIDIA Jetson Xavier NX. The JetPack SDK allows to utilize full hardware-accelerated AI-at-the-edge deployment on NVIDIA Jetson devices.

3.1.1 PyTorch

The open source machine learning framework PyTorch [65] was developed by Facebook's artificial intelligence research group in 2016 and designed for the programming language Python. PyTorch is based on the Torch library and can be combined with other popular Python libraries such as NumPy, SciPy and Cython. PyTorch allows for CPU and GPU-accelerated Tensor analysis as well as dynamically builds DNNs on a tape-based autograd system using reverse-mode auto-differentiation for gradient computation while its main advantages are characterized by flexibility and speed. Custom neural network modules may be easily created as PyTorch's Tensor Application Programming Interface (API) was designed with minimal abstractions. Furthermore, PyTorch supports Open Neural Network Exchange (ONNX) [66] enabling to export the model to be used in other machine learning frameworks and related tools based on an open standard for machine learning interoperability.

3.1.2 CUDA

CUDA [67] is a NVIDIA developed parallel computing platform, programming model and API for general computing on GPUs. The CUDA toolkit enables users to run the sequential portion of a program on the CPU while the parallel processing capabilities of all available system GPUs may be used for the computation expensive part of an application software to the full extent. Moreover, CUDA may be used with popular programming languages such as C, C++ or Python where the latter one provides an easy starting point for deep learning engineers. CUDAs ecosystem comprises GPU-accelerated libraries, a C/C++ compiler, debugging and optimization tools and a runtime library. Thus, developers are able to create, optimize and deploy high-performance applications on GPU-accelerated embedded systems such as the NVIDIA Jetson Xavier NX.

3.1.3 TensorRT

The high-performance deep learning framework NVIDIA TensorRT [68] supports the major machine learning frameworks PyTorch and Keras/Tensorflow as well as provides an ONNX parser allowing custom models to benefit from GPU hardware acceleration featured by NVIDIA's mobile platforms. The library implements a C++ and Python API, a deep learning inference optimizer as well as an optimized runtime. Hence, custom models may be created, trained and validated in an ecosystem of choice to ease the development of NN and later be deployed in terms of a TensorRT engine to achieve low latency and high throughput during inference.

3.1.4 ROS

ROS [64] is an open source framework that incorporates software libraries and tools to develop and deploy robotic applications. The framework primarily targets Ubuntu releases and provides hardware abstraction, device drivers, inter-process communication and package management. Generally, the releases of ROS can be distinguished between its two main distributions, ROS 1 and ROS 2. ROS 1 is well established and provides a centralized service for inter-process communication while ROS 2 aims at providing real-time guarantees and uses a decentralized inter-process communication system in terms of the Data Distribution Service (DDS).

3.2 Development Process

This section illustrates the chronological development process of the AttNN based dynamic object detection system and the completion of the subtasks that were necessary for the contribution of this work. Specifically, figure 3.1 visualizes the time frame for implementing the individual parts of this work and the overall progress of the thesis respectively.

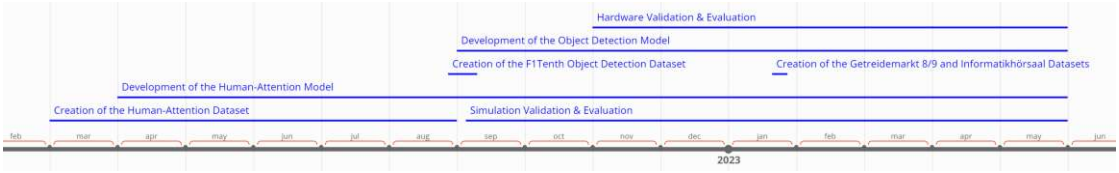


Figure 3.1: Chronological development process of the AttNN based dynamic object detection system

CHAPTER 4

Design of Att-YOLOv7

The primary focus of this thesis is two-fold and aims to formulate the problem of replicating human attention and learning a NN capable of imitating human attention on the one hand as well as relies on the design of an object detection neural network incorporating the imitated human attention on the other hand. The designed and learned AttNN attempts on imitating human attention and lays the foundation for the object detection model. In comparison with the results stated in Section 2.1, this chapter of the thesis first discusses the premises, the technical setup and the process of creating the human-attention dataset necessary for learning the human attention model, also denoted as AttNN (Section 4.1). Furthermore, the architectural design and the learning approach for the AttNN are illustrated. Section 4.2 visualizes the process of creating the object detection dataset as well as the design and learning approach for the object detection neural network. Finally, the overall structure of the AttNN based dynamic object detection system is given and the deployment in the simulation environment as well as the deployment on the F1Tenth hardware car [1] are shown in Section 4.3.

4.1 Development of the Human-Attention Model

This section provides a detailed overview on how the human-attention dataset necessary for learning the AttNN was created and how the attention filter has been designed and trained. First of all though, I would like to state that all of the work described in Section 4.1.1 that was mandatory for the creation of the human-attention dataset was done in cooperation with my colleagues Felix Resch and Stefan Ulmer.

4.1.1 Creation of the Human-Attention Dataset

For the purpose of imitating human attention in an autonomous racing context using neural networks, appropriate training, validation and test data has to be available or

acquired. The training, validation and test data has to be from a similar environmental setting and application context in comparison to the data which is provided to the model once deployed for neural networks to achieve promising results. Ideally, the given training, validation and test data also uniformly covers all the potential scenarios that may occur during operation of the model later on. In order to satisfy these requirements, we created a dataset - in the context of this thesis denoted as human-attention dataset - which focuses on the driver's attention when manually controlling F1Tenth cars [1] given various scenarios on several custom indoor tracks.

Concept & Technical Setup. The basic idea for the creation of the human-attention dataset was to simulate the navigation of the race car from the car's perspective and simultaneously record the attention of the driver. Specifically, the idea was to transmit the video stream of the car's camera over the local network, project that stream on a desktop screen and monitor the eye movements of the driver while the subject is sitting in front of the desktop monitor and manually controls the car by only using the provided camera stream. Regarding the technical setup, a variety of software and hardware components were necessary for the creation of a proper dataset. The most crucial component for the creation of the human-attention dataset was the correct tracking of the eye movements of the drivers while they navigated the car across the given track. In order to be able to record the gaze of the drivers, eye-tracking glasses called View Point System (VPS) [69] were used. Specifically, we used the VPS 19 model. The glasses communicate with an external device named Smart Unit using USB which is required for the software based interaction with the glasses. Before usage, the Smart Unit has to be used to calibrate the glasses on the pupils of the driver. Once calibrated, the Smart Unit enables the recording of the driver's eye movements using a frequency of 60 Hz and locally stores the stream of the glasses visual input as well as the x and y coordinates of the attention points for each VPS camera frame. Due to the streaming limitation of 30 Frames per Second (FPS) for the VPS camera system, multiple attention points may be recorded per frame.

In order to provide a stable and fast network stream from the car's local camera to the desktop screen of the driver, the GStreamer [70] toolchain has been used. GStreamer is an open-source multimedia framework and a NVIDIA featured software package which enables the use of hardware acceleration for NVIDIA hardware such as the Jetson Xavier NX embedded on the car. Additionally, GStreamer allows the use of multiple sinks which enables us to transmit the stream over the local network while simultaneously storing the stream on the transmitting device locally. For manually controlling the F1Tenth car, a standard console controller in combination with a ROS based joy_teleop node was used first. Due to the instability of the Bluetooth connection on greater distances and the resulting sudden unresponsiveness of the console controller, we chose to create a software-based interface linking two console controllers via WiFi which we called NetOp. Essentially, NetOp runs on a PC/Notebook, publishes the control commands of the first console controller on a specific ROS topic on which the F1Tenth car subscribes to and the control commands are directly propagated to the joy_teleop node. The second console controller is only required for enabling the manual driving mode. The latency and the

additional bandwidth over the WiFi are both negligible for our experiments as we have empirically tested in a couple of demonstrative runs. In order to acquire a smoother velocity and steering profile, we replaced the console controller linked to NetOp with a LOGITECH G923 TRUEFORCE Gaming Steering Wheel with Pedals.

The human-attention dataset is composed of multiple test runs, where one test run corresponds to one driver navigating the F1Tenth car for a certain number of laps on a given track. The data that comes along with one test run consists of all relevant frames of the car's local camera stream where the car is actually in motion as well as the set of x and y coordinates of the attention points for each of the car's camera frames which is provided in a .csv file. The frames are in RGB format and stored in the marked_frames directory of the test run. In order to assign the attention points monitored in the VPS camera space to the car's local camera space, frame matching has to be performed for which a synchronisation procedure for synchronising the local camera stream and the VPS stream had to be established. A Field Programmable Gate Array (FPGA) controlled LED array replicating a traffic light with three LEDs was programmed to successively flash each LED at a frequency of 30 Hz once a button connected to the FPGA was pressed. The frequency of 30 Hz was chosen as the car's local camera as well as the VPS were recording with 30 FPS. Therefore, the flash of the green LED - which we chose for synchronisation - was only visible on one or two frames for both camera systems denoted as synchronisation frames which were used as reference frame estimates for post-processing. Generally, this visual synchronisation procedure allowed us to synchronise the car's local camera stream and the VPS stream by triggering the synchronisation sequence in front of the car's local camera while the driver wearing the calibrated VPS was looking at the network stream on the desktop monitor before driving.

Finally, the recording of the network stream cannot be directly matched to the car's local camera frame as the transmitted network stream also covers the desktop monitor itself and its surroundings. In order to face this issue, a total of eight ArUco markers based on [71], [72] were virtually attached at the corners and edges of the transmitted network stream using the overlay function for the GStreamer receiver end. These markers are uniquely identifiable, allowing to perform an ArUco marker based display/network stream extraction on the VPS video during post-processing to detect the individual frames of the transmitted network stream for each frame of the glasses recordings. For an accurate and precise detection of the ArUco markers, the intrinsic and extrinsic parameters of the VPS camera have been estimated using a checkerboard.

Considering that no intermediate technical errors occurred at any step of the workflow, the following procedure was performed: First, the VPS was calibrated on the pupils of the driver. In a second step, the NetOp for manually controlling the F1Tenth car was launched and enabled. Afterwards, the network stream from the car's local camera was started and successful transmission to the notebook connected to the desktop monitor was tested (the receiving GStreamer application was put into fullscreen mode). The driver was required to sit in front of the desktop monitor before the VPS recording was started and the synchronisation sequence was initiated. Once the driver perceived the

green LED on the desktop monitor, he/she was allowed to drive on the given track for a certain number of laps before both recordings were stopped. Figure 4.1 depicts the interaction between the F1Tenth car and the driver.

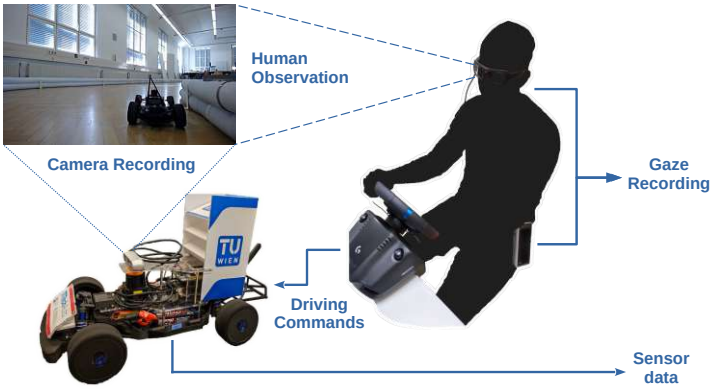


Figure 4.1: Hardware setup for recording human-gaze while manually driving a F1Tenth racecar

A recorded run consists of the car's local camera stream and the recorded VPS stream. These frame sequences have to be matched given the reference frame in both video streams such that the monitored attention point coordinates can be transformed from the VPS camera space to the car's local camera space. For the purpose of providing and learning on an efficient dataset, we are additionally merely interested in the sections of the car's local camera stream where the car is actually in motion. To this end, we used the open-source video processing software OpenSHOT [73] to post-process the car's local camera video and the VPS video, determine the respective reference frames based on the flash of the green LED in both videos as well as the frame ranges in the car's local camera video where the car was in motion. These configuration parameters are written into each run's configuration file which is further used for the post-processing pipeline.

Post-Processing Pipeline. The eye-tracking/post-processing pipeline implemented in [74] is used to successively transform the data of each recorded run into the final format for each run of the human-attention dataset. For this purpose, each run is processed consecutively while three fundamental operations are performed: the display/network stream extraction from the desktop monitor for each frame of the VPS video, the coordinates projection from the VPS camera space to the car's local camera space and the frame matching process in order to assign the correct attention point coordinates to their corresponding frame on the car's local camera space.

In a first step, the Open Source Computer Vision Library (OpenCV) [75] is used to capture the VPS video and process it frame by frame. At this point, it shall be noted that the VPS video has to be converted from .mkv to .mp4 beforehand due to some OpenCV video processing issues regarding the correct frame length of the videos. Based on the ArUco marker system, the display of the desktop monitor may be extracted from each frame of the VPS by first identifying the location of each ArUco marker in the current

frame. Once an ArUco marker has been detected, the ArUco marker library grants access to the x and y coordinates of each corner of the ArUco marker in the frame. Given the x and y coordinates of the respective ArUco markers and the capability of uniquely identifying each ArUco marker within the current frame, the display of the desktop monitor may be reliably detected by extracting the corresponding rectangle between the detected ArUco markers. Given the identified ArUco markers, this procedure may be reliably performed with at least two detected ArUco markers (diagonally opposite). The display extraction stage of the pipeline returns a .csv file holding the coordinates of the extracted rectangle for each frame of a respective run if detected.

By defining a valid attention area for each frame within the VPS video, we are able to reject all outlier attention point coordinates with respect to their corresponding valid attention areas in the second stage. The valid attention area of a given VPS frame corresponds to the extracted rectangle coordinates in the previous stage. For all valid attention point coordinates of a given VPS frame, a perspective transformation is performed. Based on the transformation matrix, the new coordinates of each attention point in the car's local camera space are computed for each frame individually. The new attention point coordinates as well as the corresponding frame number are saved in each run's transformed_coords.csv file.

In the final stage of the post-processing pipeline, the section of interest in the car's local camera video is extracted where each frame of this section is matched against its corresponding frame from the VPS domain. To this end, the frames defining the section of interest in the car's local camera space as well as the reference frames for both videos are loaded from each run's .yaml configuration file. Furthermore, OpenCV is used to capture and process the car's local camera video. For each frame of the section of interest, the corresponding VPS frame is calculated based on equation 4.1 given the defined reference frames.

$$vps_k = car_k - car_{ref} + vps_{ref}, k \in \mathbb{N} \tag{4.1}$$

vps_k refers to the VPS frame id matching the kth processed frame of the car's local camera video car_k while car_{ref} as well as vps_{ref} are the reference frames of the car's local camera video and the VPS video. In case the calculated VPS frame may not be feasible for usage as the display of the desktop monitor has not been detected or the attention point coordinates are not valid, we discard the matching frame pair. Finally, the car's local camera frames that achieved a match are stored in each run's marked_frames directory.

Human-Attention Dataset. The final dataset consists of six independent runs where a total of four different persons have been included across two different indoor track settings and different driving scenarios. These numbers only refer to valid runs where no technical issues occurred during recording and no external distraction of the drivers was given while driving. In particular, run four, six, eight and ten have been used from the track set up at Aufbaulabor recorded on 12^{th}, April 2022 while run four and five have been used from the track set up at Getreidemarkt 8/9 recorded on 23^{rd}, August 2022. For

more details on the track layouts and the environmental setting, we refer to Section 5.1.1 at this point. Two runs were recorded where the drivers had to complete the track for a couple of laps without crashing and without the presence of static obstacles (common boxes) or dynamic obstacles (F1Tenth cars). Two runs were recorded where the drivers had to complete the same track for a couple of laps without crashing and without the presence of static obstacles (common boxes) but with another F1Tenth racecar driving autonomously and in the same direction on the track (using an implementation of the follow-the-gap algorithm). The goal of the last two recorded runs was similar to the prior four runs but the driver had to complete a different track layout in a different indoor track setting. In addition to another F1Tenth car simultaneously driving on the track, static obstacles (common boxes) were placed on the track where the placement of these obstacles changed for every completed lap. Finally, four runs have been recorded while the driver was required to drive in clockwise direction while the other two runs have been recorded while the driver was required to drive in counter-clockwise direction. The human-attention dataset contains approximately 25k images and 45k attention point labels for the given images. Based on the various recorded driving scenarios, we hope that we minimized the class imbalance and bias in our attention dataset as much as possible.

4.1.2 Design and Learning of the Human-Attention Model

In order to train an artificial NN to imitate human attention in an autonomous racing context based on the collected human-attention dataset described in Section 4.1.1, an appropriate ML algorithm, training setup and representation for replicating human attention have to be defined. Above all, the general idea is that the model receives one or more camera frames and predicts the attention regions for those frames. Due to the exploratory nature of this work, two different approaches for representing human attention have been empirically evaluated. While the first and original implementation reduces the learning problem to a classification task, the second approach uses regression analysis. Considering the sequential nature of human attention and the underlying data, two learning strategies have been developed and tested. The first learning strategy is denoted as Frame based Learning (FBL) where batches of n image/frame label pairs are provided to the NN at time step t while the second learning strategy is denoted as Sequence based Learning (SBL) where batches of n sequences of k consecutive image/frame label pairs are provided to the NN at time step t.

Formally, consider input images $x_t \in \mathbb{R}^{n \times m}$, the set of attention points $A_t = \{a_{i,t}\}$ tracked by our system at time step t and attention feature maps $y_t \in \mathbb{R}^{n \times m}$ preserving the information given by A_t. Then we define the AttNN either according to equation 4.2 for FBL or according to equation 4.3 for SBL

$$\hat{y}_t = AttNN(x_t), t \in \mathbb{N} \tag{4.2}$$

$$\hat{y}_t = AttNN(\mathbf{x_t}), \mathbf{x_t} = \{x_t, ..., x_{t-k}\} \wedge t, k \in \mathbb{N} \tag{4.3}$$

where $\hat{y}_t$ represents the predicted attention feature map at time step t.

Based on the original idea, the problem of learning human attention has been formulated as a semantic segmentation problem (Section 2.4.1). We argue that this formulation is valid as we basically want to learn and infer the human-attention based RoI of the individual camera frames which are further provided as auxiliary information for the object detection NN (Section 4.2.2). Moreover, as human attention is not limited to pixel accuracy level with respect to the spatial dimensions of the camera frames that are provided as visual perceptions for the drivers, we are not interested in learning human attention at a pixel accuracy level either. Instead, the radius of the region of human attention is specified in terms of a predefined constant/hyperparameter used to generate the region of human attention for each attention point in each label. Finally, by formulation as a semantic segmentation problem, we are able to build upon state-of-the-art ML algorithms in this application domain such as U-Net [53]. This allows us to efficiently learn the region of human attention in the labels in terms of binary segmentation maps/occupancy grids. Hence, pixel-level classification is performed where every pixel in the grayscale label mask that belongs to a region of human attention is encoded as 1 whereas all remaining pixels are treated as background and are encoded as 0.

Contrarily, the second approach formulates the problem of learning human attention as an image restoration problem and applies regression analysis. This implementation is based on the premises of the previous idea but neither relies on using a predefined circle size to define the region of the human attention for each attention point in each label nor uses a binary segmentation map to encode each label. Alternatively, an isotropic 2D Gaussian distribution with a mean of 0 and a standard deviation of 1 centered at the coordinates of each attention point in each label is created. These labels are denoted as attention heatmaps and reflect the likelihood of a driver's attendance to a specific location in the image and take values in the range 0 to 255. The main advantage of this label representation is the continuous output space which enables us to apply a soft-loss for learning the attention heatmaps as adjacent pixels in the attention heatmaps are not independent from each other. On the other hand, binary segmentation discretizes the output space and assumes that every pixel in the label mask is independent from each of its neighbouring pixels.

Based on the representation of the attention heatmap labels, we are further able to encode the sequential nature of the data directly in the labels using multi-frame aggregation. For this purpose, we compute the current label by calculating the element-wise maximum of the attention heatmap of the current frame and the intensity decreased attention heatmap of the previous frame to derive the true attention heatmap at time step t. These attention heatmaps are iteratively precomputed according to equation 4.4 and equation 4.5 where y_t is the aggregated attention heatmap at time step t, h_t is the immediate attention heatmap at time step t and $(1 - r)$ is the exponential decay factor that is applied on the attention heatmap of time step $t - 1$ to account for the attention heatmap intensity variations of different time steps. Last but not least, max denotes the element-wise maximum operation.

$$y_0 = h_0 \tag{4.4}$$

$$y_t = max(h_t, (1 - r)y_{t-1}), t \in \mathbb{N} \tag{4.5}$$

To this end, this formulation of reproducing human attention aims towards allowing the model to capture the sequential nature of the data using the FBL approach without the necessity of implementing a RNN such as a LSTM as for the SBL approach.

Training & Validation

All NN architectures used for modelling the AttNN have been exclusively trained on the human-attention dataset (Section 4.1.1). The specific architectures that have been evaluated in this context are described in Section 4.1.2 in more detail. First and foremost, many drivers have been observed to sporadically focus on entities in the surrounding environment of the track while driving. These entities correspond to people present aside from the track for instance which are approximately visible in the upper third of the image from the perspective of the car's camera. Hence, all frame-attention points pairs in

the human-attention dataset are filtered based on the y location of each attention point for each corresponding image. Specifically, attention points located in the upper third of an image are dropped while frames which have no valid attention point are not considered for training or validation. Around 83.5% of the roughly 20k frame-attention points pairs are valid. The intent of this preprocessing step is to force the model to focus on locations and occurring events on the track rather than learning features from the environment. While all frames in the human-attention dataset are processed independently during training for FBL, a rolling window of size four is applied on subsequent frames during preprocessing to precompute the frame sequences on which the model is trained on for SBL. When using regression analysis, the attention heatmap labels are precomputed by iterating them once and stored in 8-bit unsigned integer (uint8) enconding in the heatmaps/ directory next to the marked_frames directory containing all images to speed up the training process. During training, the attention heatmap labels corresponding to the currently processed image batches are loaded and normalized. Several assignments of the decay factor r for the exponential decay of previous attention heatmaps have been empirically evaluated whereas the best results have been achieved by specifying $r = 0.17$ which corresponds to the visibility of the gaussian covariances for approximately 30 frames (one second). Contrarily, when using semantic segmentation, the corresponding binary segmentation masks are computed on-the-fly. For SBL, the label corresponding to the last image of the current frame sequence is utilized during training.

The spatial dimensions of the input images are resized into shape (128, 224) and the images are normalized. Given that a pretrained model is used, we additionally normalize the input images with respect to the mean and standard deviation of the dataset the models have been trained on. In this work, all pretrained models have been trained on ImageNet [76]. Therefore, we normalize all images on a per-channel mean of [0.485, 0.456, 0.406] and a per-channel standard deviation of [0.229, 0.224, 0.225] in the event of fine-tuning such a model.

The originally recorded human-attention dataset is more biased towards containing images where the F1Tenth racecar is either driving straight or taking right turns. In order to balance the distribution of observations in our human-attention dataset, we create synthetic data using data augmentation. In particular, we quadruple the size of the human-attention dataset. Initially, we apply horizontal image flipping on all images and labels of the human-attention dataset. Afterwards, we utilize colorspace augmentation by a fraction of brightness=0.4, contrast=0., saturation=0.7 and hue=0.015 to double the size of the human-attention dataset once more. Before training commences, the human-attention dataset is split into training set and validation set following a split ratio of 80% and 20%. We note at this point, that the validation set is sampled from the human-attention dataset such that it only contains samples from the original frame-attention points pairs as well as samples from the horizontally flipped frame-attention points pairs. The colorspace augmentation parameters have been selected according to the equivalent parameters specified when training YOLOv7. In order to further enhance model generalization, we rely on Exponential Moving Average (EMA) to compute the

weights of the final model based on the weighted linear combination of the EMA model's previous weights and the model's optimized weights for each iteration. Specifically, we keep track of the EMA weights according to equation 4.6

$$EMA_t = \begin{cases} \theta_0 & \text{if } t = 0 \\ (1-\alpha)\theta_t + \alpha EMA_{t-1} & \text{otherwise} \end{cases} \tag{4.6}$$

where θ_t are the optimized parameters for iteration t. The decay coefficient α is set to 0.9999 meaning that the current parameters are updated with the new parameters by a factor of 0.01%. Applying EMA on the model's parameters reduces the impact of large parameter updates for individual batches and keeps averaged model weights which tend to perform significantly better during evaluation.

We implemented Adam with decoupled weight decay (AdamW) [77] as optimization algorithm using an initial learning rate of $3e^{-4}$ and weight decay of $5e^{-5}$. These parameters have been found through hyperparameter tuning of U-Net and U-Net++ using PyHopper [78] which applies a 2-stage Markov chain Monte Carlo optimization algorithm. Based on the research proposed by [79], weight decay regularization is solely applied to the weights of convolutional or fully connected layers to improve model accuracy. Furthermore, we utilized Automatic Mixed Precision (AMP) training where each operation performed inside the model such as computing convolutions is executed in the lowest precision floating point datatype appropriate (either 16-bit floating point precision (FP16) or 32-bit floating point precision (FP32)). To prevent gradient underflow when using FP16 while optimizing inference time, gradient scaling is performed. Thus, the computed loss is multiplied by a designated scale factor. When invoking backpropagation, the magnitudes of the gradients are equally scaled preventing them from flushing to zero. Before updating the model's parameters, the gradient of each parameter is unscaled by the same scale factor to avoid interference with the learning rate. Further optimizations of the training process include using page-locked memory (memory pinning) to store data samples in order to improve host to GPU data transfer and dropping the last batch for each epoch such that only batches of batch size N are considered.

For training the model, Smooth L1 Loss according to equation 4.7 with $\beta = 1.0$ is used as loss function for regression analysis while a linear combination of Binary Cross Entropy (BCE) loss and DSC loss is applied when learning on binary segmentation masks. Both losses utilize mean reduction in order to compute the final loss for the current batch.

$$\mathcal{L}(\hat{y}, y) = \begin{cases} \frac{1}{2\beta}(\hat{y}_t - y_t)^2 & \text{if } |\hat{y}_t - y_t| < \beta \\ |\hat{y}_t - y_t| - \frac{\beta}{2} & \text{otherwise} \end{cases} \tag{4.7}$$

We compute the DSC for each batch by averaging the per sample DSC such that harder samples are weighted more during training. Moreover, we rely on a learning rate scheduler

implementing the reduce learning rate on plateau technique to ensure model convergence. In particular, the learning rate is multiplied by a factor of 0.1 if the validation metric did not change over the last two epochs. Dependent on the applied learning approach, the validation metric is expected to increase or decrease consistently for each iteration during training. The performance of models trained following regression analysis is validated based on Mean Squared Error (MSE) whereas DSC is used as validation metric for networks trained using the semantic segmentation approach. In order to objectively compare the results of both approaches on a common metric, the performance of each NN trained on attention heatmap labels is also monitored with respect to the DSC. In the validation phase, the predicted normalized attention heatmaps as well as the normalized attention heatmap labels are separately discretized based on a 50% threshold for each batch in order to further compute the DSC similar to the semantic segmentation approach.

All models have been trained and validated on a single Tesla T4 GPU with 16 GiB Video Random Access Memory (VRAM). Each model following the FBL strategy has been trained for 60 epochs using a batch size of 16 and four workers while each model following the SBL strategy has been trained for 60 epochs using a batch size of four, an image sequence size of four and four workers. Last but not least, the trained model is additionally exported as ONNX model during training such that the AttNN can further be used for the hardware deployment (Section 4.3.2).

Architectural Design & Evaluation

A selection of various network architectures from the literature proposed for image segmentation or image restoration problems such as several extensions of the standard U-Net architecture [53] have been trained on the human-attention dataset and evaluated with respect to training time, parametric complexity and DSC. Specifically, standard U-Net, Attention U-Net [55], U-Net++ [54], PAN [58], DeepLabV3+ [80], ConvLSTM U-Net and ConvLSTM Attention U-Net were evaluated in this context to elaborate which NN fits best to model the AttNN. For U-Net++, PAN, DeepLabV3+, all pretrained models and models that integrate the Concurrent Spatial and Channel Squeeze & Excitation (scSE) module we used the implementations provided by the Segmentation Models PyTorch library [81]. Except for ConvLSTM U-Net and ConvLSTM Attention U-Net, all models follow the FBL strategy and expect a normalized 4D input tensor of shape (N, 3, 128, 224) where N refers to the batch size. Each FBL model is composed of a five stage encoder-decoder network using batch normalization in the decoder. The specific architecture of the encoder backbone and the upsampling path is dependent on the respective model and not discussed in detail here. Instead, we refer to the literature for further information. In the end, a sigmoid activation function is applied on the final predictions of the model before the loss is computed. All FBL models predict a tensor of shape (N, 1, 128, 224). The labels are also provided as tensor of shape (N, 1, 128, 224). Depending on the availability of CUDA [67], all tensors are either of type FP16 or FP32.

ConvLSTM U-Net and ConvLSTM Attention U-Net follow the SBL strategy and expect a normalized 5D input tensor of shape (N, K, 3, 128, 224) where N refers to the batch size and K corresponds to the number of consecutive images per image sequence. The models consist of the same five stage encoder-decoder network as U-Net/Attention U-Net except that the fifth stage encoder-decoder subnetwork is replaced by a ConvLSTM [49]. The motivation for integrating a ConvLSTM in the U-Net architecture is two-fold in our use case. While a ConvLSTM module enables to memorize previous samples, it further internally utilizes convolutions instead of matrix multiplications, retaining the spatial dimensions of the input as opposed to a regular LSTM unit. Thus, a ConvLSTM allows to learn the spatio-temporal relation of sequential image data. The number of hidden states in our ConvLSTM module is equal to the image sequence length while the ConvLSTM cell reference implementation by [82] was adapted for this purpose. In one forward pass, the batch and sequence dimensions of the input tensor are collapsed and all images are processed by the encoder network in parallel. Before passing the extracted features to the ConvLSTM layer, the first dimension of the output of the encoder is reshaped with respect to the batch size and the image sequence length. Afterwards, all hidden states of the ConvLSTM are concatenated and simultaneously processed by the decoder network due to the symmetric design of the U-Net architecture. In order to compute the final predictions for the current batch, the N last segmentation maps are fetched from the output layer of the network. Before calculating the loss, the predictions are passed through a sigmoid activation function. Both models output a FP32 tensor of shape (N, 1, 128, 224) while the labels are again provided as FP32 tensor of shape (N, 1, 128, 224). The usage of the FP16 data type for all tensors during training again depends on the availability of CUDA.

We performed an exhaustive analysis on different NN algorithms. Next to the base models trained from scratch, ImageNet pretrained variants of U-Net and U-Net++ using different encoder backbones were trained. Furthermore, the integration of the scSE module in the decoder of several models has been evaluated. Figure 4.2 compares the training and validation losses as well as the training and validation DSC scores of all validated model architectures throughout all epochs. On the other side, table 4.1 provides detailed statistics about the training time (in hours), parametric complexity and validation DSC. Complementary, figure 4.3 visually relates training time and validation DSC whereby the marker size indicates the approximate number of model parameters. In order to further compare models trained using regression analysis or semantic segmentation, several architectures have been trained and evaluated using both approaches. Models trained using the classification approach are marked with the suffix *CLS* accordingly. All models have been trained from scratch and utilize multi-frame aggregated human-attention heatmap labels if not stated otherwise.

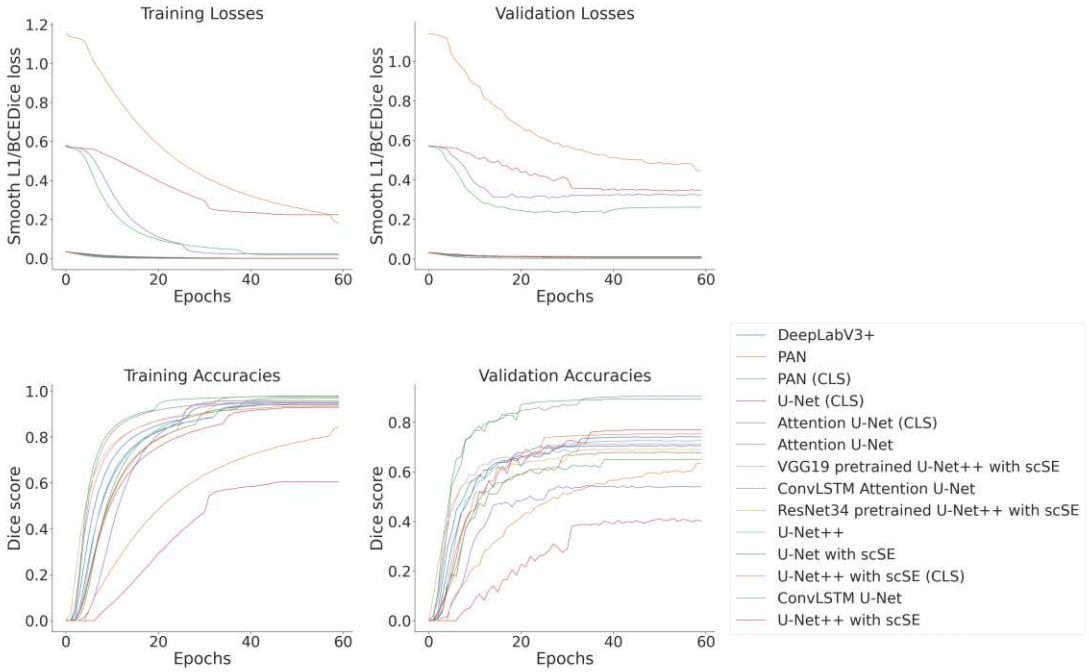


Figure 4.2: Illustration of the training statistics of each model

Model	Training time [h]	Parameters	Validation DSC
DeepLabV3+	5.58	22 437 457	73.90
PAN	5.49	21 475 816	74.67
PAN (CLS)	5.67	21 475 816	64.61
U-Net (CLS)	13.80	34 520 193	40.63
Attention U-Net (CLS)	15.19	34 870 761	56.57
Attention U-Net	15.19	34 870 761	69.03
VGG19 pretrained U-Net++ with scSE	27.84	40 950 297	70.03
ConvLSTM Attention U-Net	63.04	27 524 327	90.50
Resnet34 pretrained U-Net++ with scSE	11.28	26 281 137	67.98
U-Net++	7.35	26 078 609	72.95
U-Net with scSE	5.45	24 436 369	71.17
U-Net++ with scSE (CLS)	11.53	26 281 137	61.74
ConvLSTM U-Net	57.15	27 437 185	91.49
U-Net++ with scSE	11.77	26 281 137	76.68

Table 4.1: Characteristic comparison of model architectures

Based on the results provided in table 4.1, we selected ConvLSTM U-Net as the architecture for the AttNN as ConvLSTM U-Net achieves the highest DSC score on the validation set. While the difference in DSC score between ConvLSTM U-Net and Con-

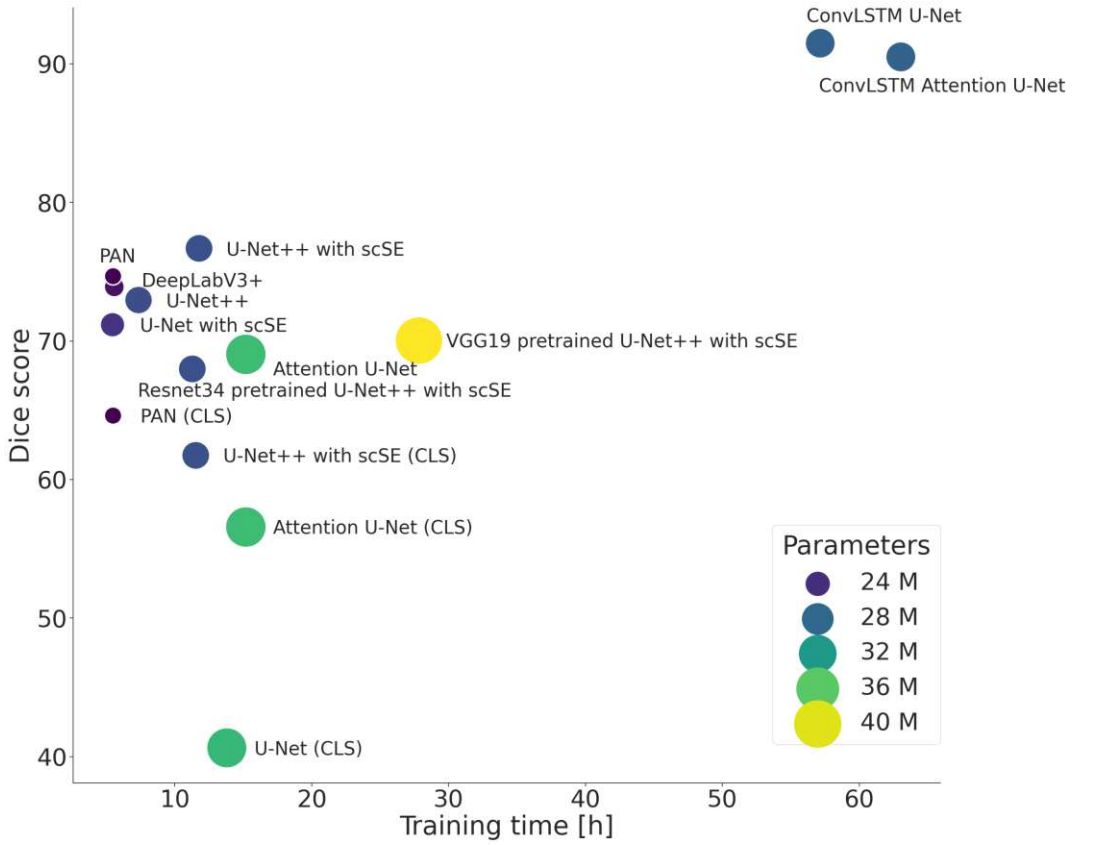


Figure 4.3: Comparison of model architectures

vLSTM Attention U-Net on the validation set is negligible, both models outperform models trained using FBL by a large margin. In particular, ConvLSTM U-Net exceeds U-Net++ with decoder attention type scSE which represents the best model using FBL and regression analysis in DSC score on the validation set by 14.81%. Conversely, PAN (*CLS*) represents the best model using FBL and semantic segmentation, where we note a remarkable gap of 26.88% in DSC score on the validation set compared to ConvLSTM U-Net. Considering the performance of pretrained, fine-tuned models and models trained from scratch, we observe that the U-Net++ with decoder attention type scSE trained from scratch is superior in performance compared to the ImageNet pretrained VGG19 backboned U-Net++ with decoder attention type scSE by 6.65% in DSC score. Likewise, U-Net++ with decoder attention type scSE outperforms the ImageNet pretrained ResNet34 backboned U-Net++ with decoder attention type scSE by 8.7% in DSC score.

Our empirical evaluation of models trained with FBL and multi-frame aggregated attention heatmap labels and models trained with FBL and binary segmentation mask labels further shows that human-attention feature maps represented as time-variant gaussian

heatmaps are more efficient for NN to learn. The importance of the human-attention feature map representation is evident when comparing the DSC score on the validation set for models trained using both learning strategies. For instance, U-Net++ with scSE achieves a 15% higher DSC score on the validation set when representing human-attention feature maps as multi-frame aggregated gaussian heatmaps as opposed to binary segmentation masks. We may also theoretically support this observation as human attention is a time and stimuli sensitive biological mechanism which is better approximated by encoding human-attention feature maps as time-variant gaussian heatmaps.

Finally, we present an abstract schematic of our human-attention model in figure 4.4, visualizing the input and output layer as well as the major components of the NN architecture.

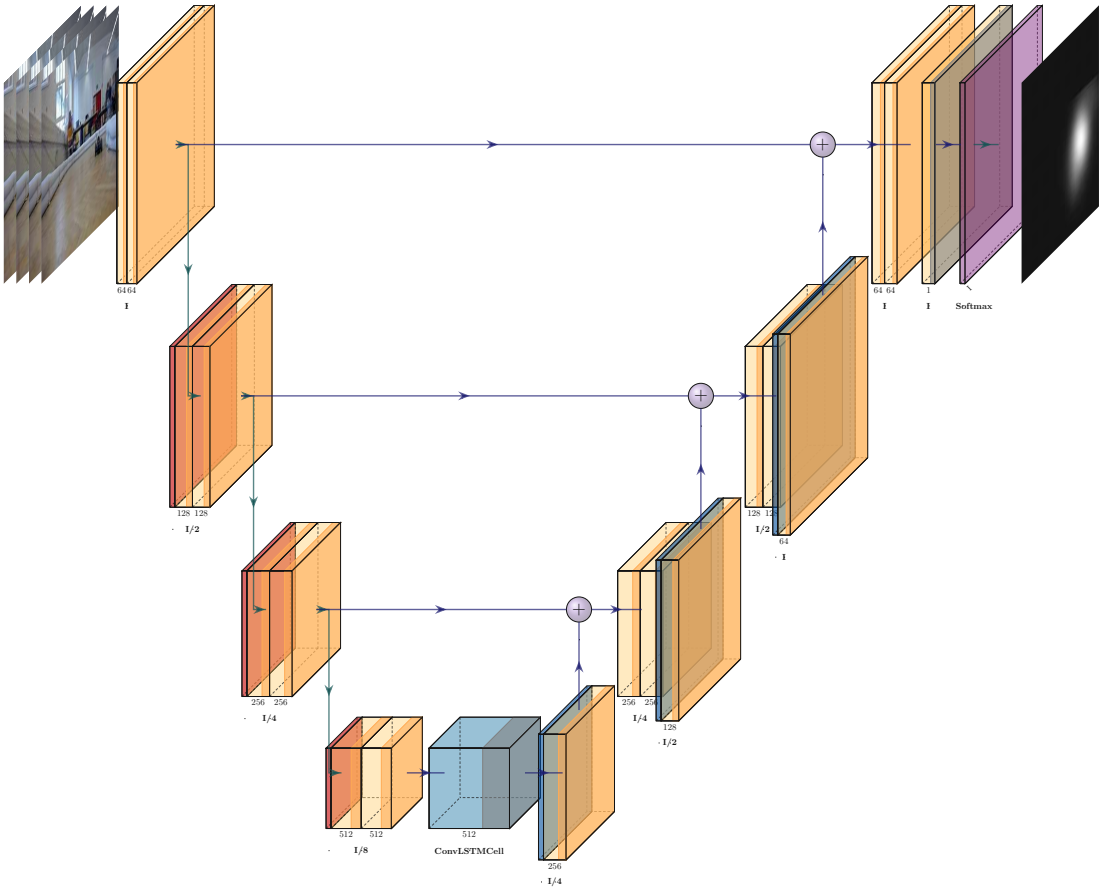


Figure 4.4: Abstract architecture of the AttNN

4.2 Development of the Object Detection Neural Network

For the purpose of performing reliable high-speed inference in object detection, a ML-based object detection algorithm capable of satisfying real-time requirements and inducing low latency is mandatory. To this end, the focus of this section relies on the creation of the 2D object detection dataset which is described in Section 4.2.1 as well as the design, training and validation of the object detection model as discussed in Section 4.2.2.

4.2.1 Creation of the F1Tenth Object Detection Dataset

In an autonomous driving context, a vast variety of different objects exists where a HAD agent has to locate each object in the environment and predict their temporal behaviour dependent on their category. Especially in an autonomous racing context, the localisation of the opponent and other objects on the track and the prediction of their temporal behavior is highly important in order to perform evasive actions or attempt on overtaking at the right moment. Generally, objects on the track may roughly be categorised into static and dynamic obstacles. Static obstacles have a fixed position on the track while dynamic obstacles are objects moving relatively to the agent. In particular in F1Tenth [1], the objects of interest on the track are represented by two object categories: F1Tenth cars and common boxes. Specifically, a F1Tenth car refers to a custom build of a F1/10 scale Formula 1 race car as illustrated in [1] while cuboid boxes with a TU Vienna logo on each side are used as common boxes. For the simulation deployment (Section 4.3.1), the hardware deployment (Section 4.3.2) as well as the overall evaluation of the AttNN based dynamic object detection system and the comparison to the state-of-the-art in computer vision research (Section 5), an object detection dataset consisting of these two object classes is considered.

Analog to the creation of the human-attention dataset in Section 4.1.1, the training, validation and test data for the object detection dataset, also denoted as F1Tenth object detection dataset needs to be from an equivalent environmental setting as the input data provided to the object detection NN once deployed. Therefore, several videos of different runs across two distinct indoor track settings initially recorded for the human-attention dataset have been post-processed. In particular, the frames of each selected run's video that contain objects of interest have been extracted and manually annotated such that each object of interest in a selected frame is marked with its corresponding class label and an enclosing 2D bounding box. Finally, between 8% and 10% of the number of labelled images have been chosen from the remaining non-selected frames of the individual runs and are used as background images/negative samples in the object detection dataset. As a consequence, the number of false positives of the object detection NN shall be significantly reduced. In summary, the F1Tenth object detection dataset contains a total of 7819 images and 7303 labels. The split between training, validation and test data is 70%, 15% and 15% respectively. The dataset covers three major scenarios: background images, single class images and multi class images. Table 4.2 provides an overview of the number of used instances per category as well as the number of occurrences of each of

the two classes for training, validation and test data.

Category	Training	Validation	Test
background images	736	155	166
f1tenth car	3807	832	814
box	1298	276	285
single class	4421	952	934
multi class	315	66	74

Table 4.2: F1Tenth object detection dataset statistics

Finally, the F1Tenth object detection dataset consists of RGB frames that have been recorded using a Logitech Webcam and an Intel RealSense camera mounted in front of the F1Tenth car while driving. Due to the insufficient consistent fixation of the camera system while the car was in motion, the background noise present in the camera frames has been captured in the F1Tenth object detection dataset ensuring a similar noise distribution when deploying the AttNN based dynamic object detection system on the physical car.

4.2.2 Design and Learning of the Object Detection Neural Network

High-speed inference and low-latency for object detection algorithms operating in time-critical and highly dynamic environments is crucial when deploying the model on a cyber-physical system such as a F1Tenth racecar. Building upon the latest research in the domain of computer vision, the recent advancements in vision-based object detection NN culminate in the one-stage object detector YOLOv7 [29]. YOLOv7 outperforms most state-of-the-art object detection algorithms on the PASCAL VOC/COCO datasets in terms of inference time and mAP including one-stage object detectors such as SSD [37] and the predecessors of YOLOv7 [28, 32, 33, 34, 35] as well as two-stage object detectors such as R-CNN [38], Fast/Faster R-CNN [39, 40] and the end-to-end object detection with transformers architecture denoted as DETR [4]. Based on its impressive results, YOLOv7 has been chosen as the object detection NN for this thesis. The reference PyTorch implementation adapted in this work is provided by [83].

The general breakdown of the architecture of the YOLO family, the idea behind the YOLO object detection NN as well as the algorithm of an advanced YOLO variant, namely YOLOv4 [34] have already been presented in Section 2.4.2. The latest instance of YOLO - YOLOv7 - builds upon the algorithms of its predecessors YOLOv4 and YOLOv5 [35] and defines several improvements which are integrated in its new architectural design and training routine. At this point, we intend to provide a more detailed discussion on the improvements of the YOLOv7 architecture and its training procedure.

Most commonly, researchers aim to optimize the architecture of a NN by reducing the number of parameters, the computational density of the model or the total number of computations. The length of the longest path a gradient has to traverse during

backpropagation has a huge impact on the learning capabilities of a network, especially in terms of expressiveness of the model and learned feature diversity. For DNNs to learn efficiently and converge fast, Efficient Layer Aggregation Networks (ELAN) has been proposed which allows to control the shortest and the longest gradient path in the network. However, by rapidly increasing the number of computational blocks, the utilization rate of the parameters in the network may decline. To resolve this issue, Wang et. al. proposed Extended Efficient Layer Aggregation Networks (E-ELAN) which changes the architecture of the computational blocks by applying expand, shuffle and merge cardinality to continuously enhance the network's learning capability. Contrarily, the architecture of the transition layer is left unchanged, enabling the network to maintain its original gradient path. More specifically, E-ELAN implements grouped convolutions to expand the channel dimension and cardinality of each computational block as well as uses the same channel multiplier and group parameters on all feature maps of a computation layer. Finally, the feature maps will be shuffled, concatenated and added to the feature vector from the previous stage to perform merge cardinality and compute the feature vector of the current stage.

In order to meet the versatile requirements of different applications on inference speed, memory consumption and utilisation as well as mAP, concatenation based computer vision models are usually designed and trained for multiple scales as the scaled depth results in a different number of layers and the scaled width leads to a different number of channels in the network. To overcome this limitation and adapt the initial design for multiple use cases, the authors of YOLOv7 use a compound scaling method which computes the change in the output kernel dimensions and scales the width of the network correspondingly. Hence, the network maintains an optimal architecture and its properties for an already designed algorithm while scaling depth, width and resolution of the network.

In contrast to enhancements in the architectural design, improvements in the training strategy of YOLOv7 which increase the overall accuracy of the algorithm while maintaining or reducing inference time are denoted as trainable BoF. YOLOv7 introduces and utilizes two new techniques: planned re-parameterized convolution and coarse for auxiliary and fine for lead loss.

The general idea behind re-parameterization techniques is to increase model robustness and its ability for generalisation by averaging a set of model weights of multiple models trained on different training data or across distinct epochs. Recent research focused on module level re-parameterization where individual computational blocks or layers of the network are transformed based on specific re-parameterization strategies e.g. merging of multiple layers. The authors of YOLOv7 used gradient flow propagation paths in order to optimally combine re-parameterized convolutions with different network structures.

The second trainable BoF is based on the concept of deep supervision where especially for very deep NN additional information is provided by inserting an auxiliary head in the middle layers of the network. Via communication between the auxiliary head and the lead head which is responsible for the prediction of the final output of the network,

the auxiliary head is able to build upon knowledge that the lead head has learned, thus enabling the lead head to focus on learning new information. In order to generate appropriate labels for both heads, YOLOv7 proposes a coarse-to-fine lead head guided label assigner. The soft labels for the lead head and the auxiliary head are computed based on the ground truth and the prediction of the lead head. However, the constraints of the coarse soft labels for the auxiliary head are weakened as the auxiliary head has a lower learning capability than the lead head. Hence, an additional number of false positives in the surrounding grid cells is accepted as correct prediction for the auxiliary head.

Further adaptions of the network's structure and the training process have been performed such as integrating batch normalization between the convolution layer and the activation layer as well as using implicit knowledge in You Only Learn One Representation (YOLOR) [36] and convolution feature maps to reduce the number of computations at inference time. Finally, YOLOv7 applies an EMA to average the weights of the model ensemble in order to arrive at the final model's weights, a technique also referred to as the Mean Teacher method [84].

Architectural Design, Training & Validation

In terms of the specific architecture used for the YOLOv7 models in this thesis, we differentiate between the YOLOv7 base model (**Native YOLOv7**) and the Attention as feature based YOLOv7 algorithm (**4-channel YOLOv7**). The native YOLOv7 algorithm is applied to perform inference on full-scale input images and is used as evaluation baseline for the performance benchmark comparison later on in Section 5. While 4-channel YOLOv7 also performs inference on full-scale images, the model leverages the predicted human-attention heatmaps for the full-scale input images as additional channel information for the YOLOv7 base model to increase feature diversity and robustness. Hence, the primary architectural difference of the two networks relies in the number of input channels as well as in the number of trainable parameters.

The input of all YOLOv7 networks is a normalized FP32 tensor of shape (N, C, H, W) where N is the batch size, C is the number of channels and H and W are the height and width of the input images respectively. In detail, native YOLOv7 expects a normalized FP32 tensor of shape (N, 3, 384, 640) while 4-channel YOLOv7 expects a normalized FP32 tensor of shape (N, 4, 384, 640). The size of the spatial dimensions of the input tensors results from the letterbox format computed by YOLOv7 when performing rectangular inference on the images of the F1Tenth object detection dataset for a selected target image width of 640 pixels. For all experiments conducted throughout this thesis, all object detection models resize input images such that the target image width corresponds to 640 pixels respectively. Based on the utilized anchor box configuration, the output of the YOLOv7 networks is a FP32 tensor of shape (15120, 7) while the final output after NMS is a FP32 tensor of shape (K, 7) where K is the number of 2D bounding box predictions for the given input image with an IoU score greater equal to the IoU

threshold specified for NMS. For all models, the IoU threshold has been set to 0.20 for training and 0.65 for evaluation later on.

All instances of YOLOv7 have been exclusively trained on the F1Tenth object detection dataset (Section 4.2.1) from scratch. Hence, no pre-trained models have been applied for training. Specifically, the images and labels located in the /images/train and /labels/train directories have been used for training. For model validation and evaluation later on, the images and labels from the /images/val, /labels/val and /images/test, /labels/test directories have been used. Generally, the proposed training routine as well as the hyperparameters that were stated by the authors of YOLOv7 for learning their network's weights have also been used to train all instances of YOLOv7 for this work. In particular, a series of data augmentation techniques were implemented to increase the robustness and generalization of the YOLOv7 model. We apply horizontal image flipping with a probability of 0.5, image translation by a fraction of +/-0.2, image scaling by a gain of +/-0.5, HSV-colorspace augmentation by a fraction of h=0.015, s=0.7 and v=0.4 as well as the novel image mosaic data augmentation with a probability of 0.8 for four image mosaics or a probability of 0.2 for nine image mosaics during training. Moreover, YOLOv7 implements EMA for computing an averaged set of weights across all iterations with a decay coefficient of 0.9999. The latest EMA weights are iteratively calculated based on the weighted linear combination of the previous EMA weights and the updated weights of the model of the current epoch.

In our implementation, SGD is applied as optimizer with an initial learning rate of $1e^{-2}$, a momentum of 0.937 and weight decay of $5e^{-4}$. For validating YOLOv7 in terms of mAP, three different metrics have to be assessed resulting in a compound loss function. The score computed by the loss function is derived from the weighted linear combination of an objectness loss, a bounding box or regression loss and a classification loss. The weight coefficients are 0.7, 0.05 and 0.3 respectively. In detail, the overall loss is computed by initially targeting center prior anchor boxes for each FPN head. Those priors are refined using the Optimal Transport Assignment (OTA) algorithm which formulates label assignment as a global optimisation problem for each image and solves common issues regarding occlusion or multiple adjacent objects when using IoU. The goal of the OTA algorithm is to find an optimal predicted bounding box to target bounding box label assignment minimizing the total cost for each image. While solving the global optimisation problem is computationally expensive, YOLOv7 implements a simplified version of OTA by finding a matching target bounding box for each of the n predicted bounding boxes having the lowest cost. Once the label assignment has been computed, the objectness loss is determined through the BCE loss of the predicted objectness probability and the CIoU between the predicted bounding boxes and the matched targets. The bounding box loss is computed for all proposed anchor boxes and their corresponding targets based on the mean of the inverse probability of CIoU. The classification loss is obtained by assessing the BCE loss between the predicted class probabilities of each anchor box and a one-hot encoded vector representing the ground truth class index of the matched target bounding box. In case auxiliary heads are integrated into the model,

each loss component is added to the respective loss component of the lead head w.r.t. to the contribution weight of the auxiliary head. Similarly, the objectness loss is scaled by the weight of the FPN head. Then, the true loss score is defined by the batch size multiplied by the sum of the weighted linear combination of the objectness, regression and classification losses. Finally, YOLOv7 utilizes AMP training.

Analog to the authors of YOLOv7, we apply a cosine learning rate scheduler to adjust the learning rate during training with a linear warmup of three epochs. Additionally, the optimizer only applies weight decay regularization to parameters of convolutional or fully connected layers based on the research proposed by [79]. The weight decay is also scaled based on the used batch size during training where the base value is fixed for batch sizes lower than the nominal batch size and linearly increased for batch sizes larger than the nominal batch size (here 64). Moreover, YOLOv7 implements gradient accumulation such that the optimizer does not update the parameters of the model after computing each batch, but after a designated number of iterations. Both techniques tackle the issue of training the network with inconsistent batch sizes which might otherwise result in unstable training or require a different selection of hyperparameter values.

Eventually, the selection of appropriate anchor box sizes is fundamental for YOLOv7 to perform well on the trained dataset as candidate anchor boxes are generated based on the provided anchor box sizes at each anchor point. The anchor size configuration for each of the FPN heads we used is based on the original YOLOv7 implementation for the COCO dataset and given in the following:

$$\text{- } [12, 16, 19, 36, 40, 28] \ \#P3/8$$

$$\text{- } [36, 75, 76, 55, 72, 146] \ \#P4/16$$

$$\text{- } [142, 110, 192, 243, 459, 401] \ \#P5/32$$

While native YOLOv7 can be trained directly by providing the target dataset and the target hyperparameter configuration, various modifications to the dataloader of YOLOv7 had to be made for training 4-channel YOLOv7. In order to train 4-channel YOLOv7 on the RGB input images as well as on the corresponding attention heatmaps, the dataloader has been modified such that the associated attention heatmaps are simultaneously processed for each image batch. For this reason, the AttNN was used to predict the attention heatmaps for the images in the F1Tenth object detection dataset. The predicted attention heatmaps have been stored in the /heatmaps/train, /heatmaps/val and /heatmaps/test directories accordingly. Additionally, data augmentation techniques that are applied to input images and geometrically alter the original images (except for HSV-colorspace augmentation) have to be equally applied to the corresponding attention heatmaps of those images to compute consistent image/attention heatmap pairs. Many of the applied data augmentation techniques such as image mosaic augmentation and random affine transformation perform sampling operations, either for computing additional indices in the case of mosaic augmentation or for calculating the new RoI in the

case of random affine transformation. To ensure consistency between the input images and the corresponding attention heatmaps, we initially perform the sampling operation and afterwards simultaneously compute the augmented images/attention heatmaps using the sampled configuration. The final input of the 4-channel YOLOv7 network results from the concatenation of the image/attention heatmap pairs along the channel axis where the fourth channel contains the corresponding attention heatmap.

Evaluation of YOLOv7 model variants

All YOLOv7 models have been trained on a single Tesla T4 GPU with 16 GiB VRAM for 300 epochs using a batch size of 16 and four workers and perform inference on rectangular images of size 384×640. During preprocessing, the original images of size 480×848 are resized and padded if needed according to the letterbox format expected by YOLOv7 for rectangular inference. We selected a target image width of 640 pixels for comparing our models running rectangular inference. Figure 4.5 illustrates mAP@.5 and mAP@.5:.95 scores of native YOLOv7 and 4-channel YOLOv7 on the training and validation set of the F1Tenth object detection dataset across all epochs. Moreover, table 4.3 compares training time (in hours), parametric complexity, precision (P), recall (R), mAP@.5 and mAP@.5:.95 scores of both models on the corresponding validation set. We compare the object detection accuracy of all models in this thesis based on mAP, as this is the defacto standard metric for evaluating object detection algorithms. Notably, mAP is defined as the area under the precision-recall curve where mAP@.5 denotes mAP evaluated on an IoU confidence threshold of 0.5 while mAP@.5:.95 denotes the average mAP score when evaluating mAP on an IoU confidence threshold of 0.5 up to 0.95 with a step size of 0.05 individually. Finally, the precision, recall, mAP@.5 and mAP@.5:.95 scores represent the average value across all classes.

Model	Training time [h]	Parameters	P	R	mAP@.5	mAP@.5:.95
Native YOLOv7	23.827	37 201 950	0.998	0.998	0.998	0.967
4-channel YOLOv7	25.92	37 202 238	0.997	1	1	0.963

Table 4.3: Characteristic comparison of YOLOv7 variants

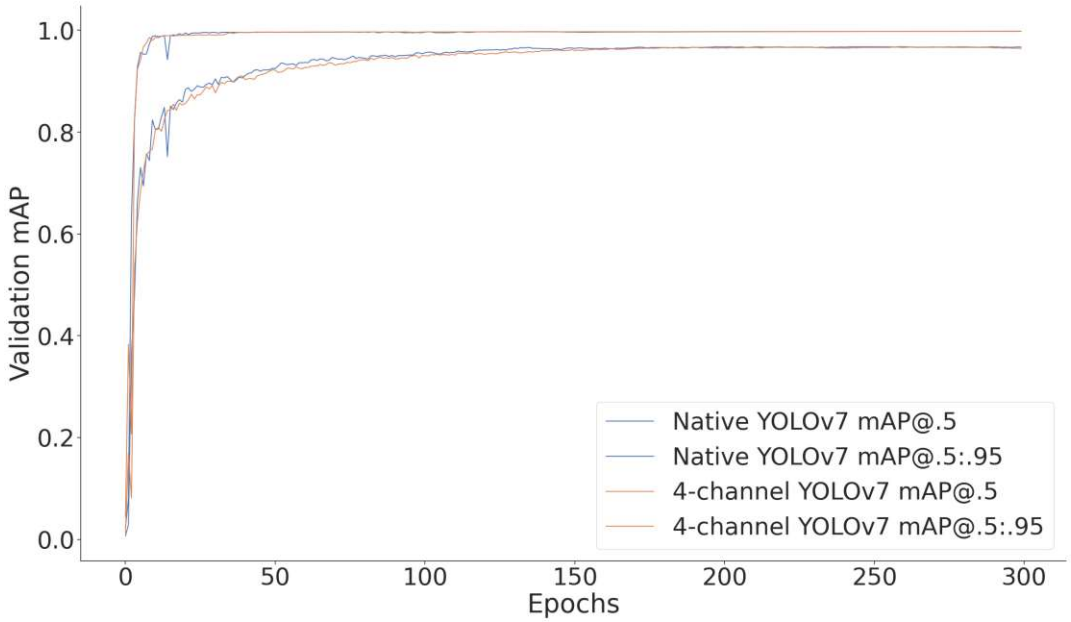


Figure 4.5: mAP scores of each YOLOv7 instance on the validation set

4.3 Algorithm and Deployment of the System

In order to deploy and evaluate the AttNN based dynamic object detection system, the trained attention filter (Section 4.1) and the trained object detection models (Section 4.2) have to be efficiently integrated. Essentially, this work is based on the premises that every visual location a human driver attends to while steering a F1Tenth car has to be of interest and most importantly, a human driver will eventually focus and lay his/her attention on any static or dynamic objects in the environment in order to safely navigate the F1Tenth car and avoid a crash. For this reason, we propose and evaluate our novel approach, denoted as AttNN based dynamic object detection system (**Att-YOLOv7**). Att-YOLOv7 incorporates 4-channel YOLOv7 as object detection model which utilizes the predicted human-attention feature maps by the AttNN as additional channel information with the goal of providing a more robust and generalized YOLOv7 model. Figure 4.6 outlines the overall pipeline of the AttNN based dynamic object detection system. We further discuss our approach and provide details on how we evaluated the performance in simulation in Section 4.3.1. Finally, Section 4.3.2 gives insights on how we deployed our system on the F1Tenth racecar, performed optimizations and the communication with HAD agents.

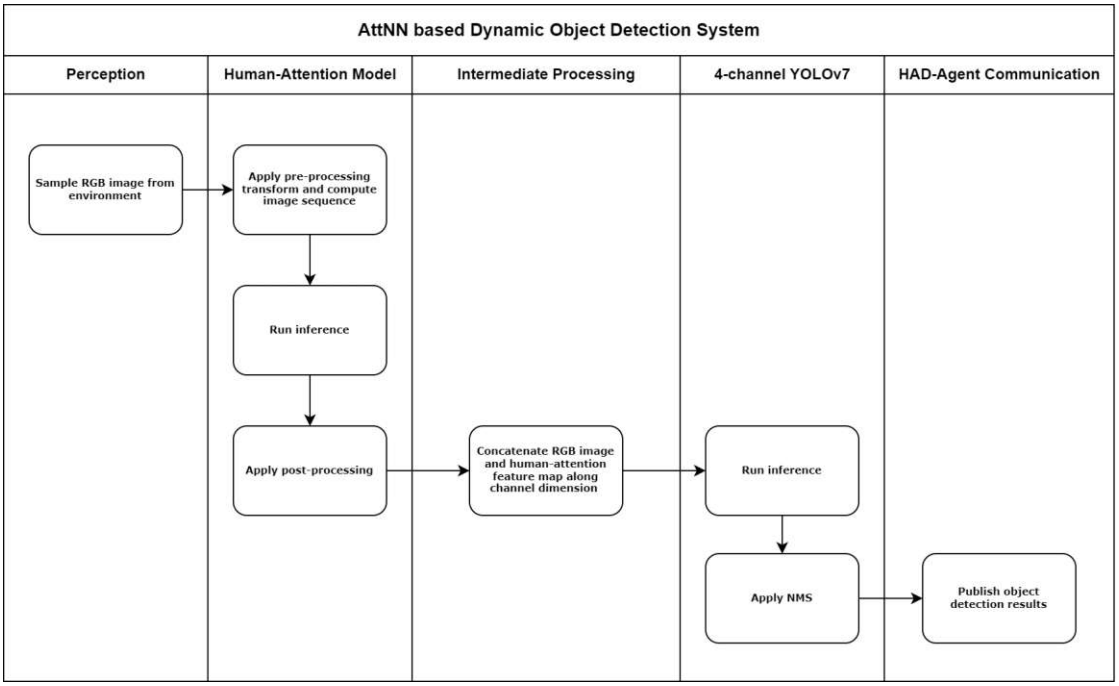


Figure 4.6: Sequence diagram of Att-YOLOv7

4.3.1 Simulation Deployment

The overall system layout of the AttNN based dynamic object detection system given in figure 4.6 was implemented in a simulation environment using the toolchain as described in Section 3.1. When comparing the simulation deployment of our system to the deployment on the real F1Tenth racecar (Section 4.3.2), minor modifications to the pipeline illustrated in figure 4.6 had to be made. The interface to the camera is replaced by a custom dataloader instance which allows to iterate through a selected directory containing a given set of test images. The post-processing stage of the pipeline has also been altered for simulation as the predictions are not communicated to the HAD agent but are instead visualized on the provided input images. The final inference results are stored in a designated directory containing all input images marked with the corresponding 2D bounding box predictions, class name and confidence score while the color of the predictions represents the predicted class. For evaluating the performance of our approach on relevant metrics such as mAP, we accumulate the score for the respective metrics computed on the inference results of each image and derive the final values by averaging over the entire dataset. We equivalently compute the mAP score for all object detection baselines in Chapter 5.

For the purpose of generating and storing the current frame sequence required as input for the human-attention model, a ring buffer is initialized with the size of the image sequence length k and k copies of the first frame retrieved from the input source e.g. from a RGB camera or a custom dataloader. For every iteration, the oldest frame in the

ring buffer is replaced with the most recent one and the entire frame sequence is returned with the most recent frame being located at index $k-1$. Independent of the deployment method, this ring buffer implementation is used to capture the sequential nature of our application domain in a simple way.

Expanding on the algorithmic details, Att-YOLOv7 exploits the predicted human-attention heatmap computed by the AttNN as additional attentional feature for YOLOv7 to retrieve supplementary information for learning and infering the 2D bounding box and class labels in each scene. In particular, the channel depth of the input layer of the YOLOv7 network is extended by adding an auxiliary fourth channel that represents the non-normalized human-attention feature maps as uint8 grayscale images for the corresponding RGB input images. Values significantly larger than zero in the human-attention feature map indicate regions of predicted human-attention. Before inference, the human-attention feature map is normalized along the given RGB image. In order to concatenate the human-attention feature map/RGB image pairs, the human-attention feature maps are resized w.r.t. the spatial dimensions of the input images after computing the letterbox format for the original input images given the selected target image width of 640 pixels.

4.3.2 Deployment on the F1Tenth research platform

Apart from the robustness evaluation of Att-YOLOv7 on OOD data, the feasibility of our approach for real-world autonomous driving/racing applications shall also be assessed and validated. Therefore, we deployed our system on a custom build of a physical F1Tenth racecar.

The custom build of our F1Tenth racecar is primarily based on the standard F1Tenth hardware components according to the F1Tenth documentation [1]. Minor modifications have been made regarding the placement of the individual hardware parts. Furthermore, we connected a visual camera system to the NVIDIA Jetson Xavier NX where initially a Logitech Webcam and later on an Intel RealSense camera mounted above the front axle of the F1Tenth car has been integrated. The specific software stack used on the F1Tenth research platform has already been described in Section 3.1.

In order to deploy the AttNN based dynamic object detection system efficiently on the target platform regarding execution time and to enable an easy collaboration with HAD agents, the pipeline illustrated in figure 4.6 has been implemented in a ROS node and the TensorRT Python API has been used to optimize the frame rate of the system by running the AttNN as well as 4-channel YOLOv7 on the GPU. For this purpose, the weights of the AttNN and 4-channel YOLOv7 models have been converted to ONNX format using ONNX opset version 12 with constant folding. The ROS node either expects ONNX models as input and compiles the networks into FP32/FP16 TensorRT engines or directly runs supplied TensorRT engine files on the GPU. Two operational modi are supported which can be configured using the parameters in the corresponding launch file of the ROS node; running native YOLOv7 or running Att-YOLOv7.

Furthermore, NVIDIA Jetson Xavier NX supports 8-bit integer (INT8) model inference directly on the hardware, which substantially increases model throughput, reduces inference latency and memory consumption while slightly dropping in mAP compared to models using FP32 or FP16. The ROS node implements a custom INT8 calibrator class for the AttNN as well as for all variants of YOLOv7 used in this thesis. Specifically, we apply Post-Training Quantization (PTQ) based on the workflow stated in the TensorRT documentation. For running inference using an INT8 calibrated pretrained model using PTQ, a calibration dataset for the respective model has to be supplied. As TensorRT uses the calibration dataset for INT8 quantization of the model's parameters, we separately use the images from the /images/val directory of the F1Tenth object detection dataset for calibrating the AttNN as well as 4-channel YOLOv7 to avoid overfitting on the training data of each model. In particular, we apply a calibration size of 1024 image samples for calibrating each model individually. The same dataset and settings are used for calibrating native YOLOv7.

Last but not least, our system enables HAD agents to instantaneously react to detected objects in the environment. The system directly runs inference on the frames perceived by the connected camera device and publishes object detection predictions on the /hardware_inference/detections ROS topic. In particular, object detection predictions are published as messages of type vision_msgs/Detection2DArray where each detected object is represented in YOLO format and defined by the vision_msgs/Detection2D message type. The bounding box coordinates are non-normalized and stored as vision_msgs/BoundingBox2D message type. On the opposite, class id and confidence score are represented using the id and score fields of the vision_msgs/ObjectHypothesisWithPose message type. To this end, HAD agents may use our perception system to enhance the efficiency and safety of their motion-planning and control algorithms.

CHAPTER 5

Evaluation of Att-YOLOv7

This chapter of the thesis presents the main results of comparing our proposed AttNN based dynamic object detection system with different state-of-the-art object detection baselines in computer vision research. Specifically, this work is evaluated against YOLOv7, Faster-R-CNN and DETR where we conducted various experiments in terms of inference cost, model robustness and mAP. The primary focus of our experiments aims to show the robustness of our proposed AttNN based dynamic object detection system on OOD data, in particular on illumination perturbations in the input data. For computer vision-based object detection NN, high resilience to changes in illumination is crucial as these models are generally sensitive to the illumination conditions in the environment which are not controllable in autonomous driving/racing. To this end, we show that human attention may potentially provide a way for resolving the illumination vulnerability of computer vision-based object detection models used for autonomous racing/driving tasks. Finally, in Section 5.2, the limitations of our approach are discussed, the feasibility of deploying the AttNN based dynamic object detection system on hardware is evaluated and the inference results obtained on the real F1Tenth racecar are compared to the results from the preceding simulation evaluation.

5.1 Single-Agent/Multi-Agent Evaluation

In this section, we present the main contributions of this work. We conducted several experiments regarding the evaluation of our proposed AttNN based dynamic object detection system and the selected object detection baselines on OOD data. Section 5.1.1 describes the training of Faster R-CNN and DETR on the F1Tenth object detection dataset as well as the recording of additional data necessary for performing various experiments on distinct OOD data. In Section 5.1.2, we present the results of comparing our proposed system with native YOLOv7, Faster R-CNN and DETR on OOD data, in particular on illumination disturbed inputs of three different datasets. Finally, we

provide a visual comparison between the human-attention based feature maps generated by the AttNN and the artificial machine attention utilized by the transformer of the DETR architecture in Section 5.1.3.

5.1.1 Experimental Setup

In order to evaluate and compare our proposed system with state-of-the-art object detection baselines in computer vision, the selected algorithms have to be trained on the F1Tenth object detection dataset. In contrast to YOLOv7, Faster R-CNN and DETR expect the underlying dataset to be in PASCAL VOC format or COCO format accordingly. While the directory structure for storing the images and labels in the proper subsets is different between YOLO, PASCAL VOC and COCO, the specific representation of the annotations also deviate. PASCAL VOC expects the labels to be stored in Extensible Markup Language (XML) format whereas COCO assumes JavaScript Object Notation (JSON) encoding. However, the major difference relies in the representation of the bounding box annotations. YOLO expects the format $(x_{center}, y_{center}, width, height)$ for describing bounding box labels and normalized values in range [0, 1]. Alternatively, PASCAL VOC stores bounding boxes as $(x_{min}, y_{min}, x_{max}, y_{max})$ whilst these are represented as $(x_{min}, y_{min}, width, height)$ for COCO. Both formats assume non-normalized coordinates. For more details on the stated dataset formats and differences, we refer to the literature at this point.

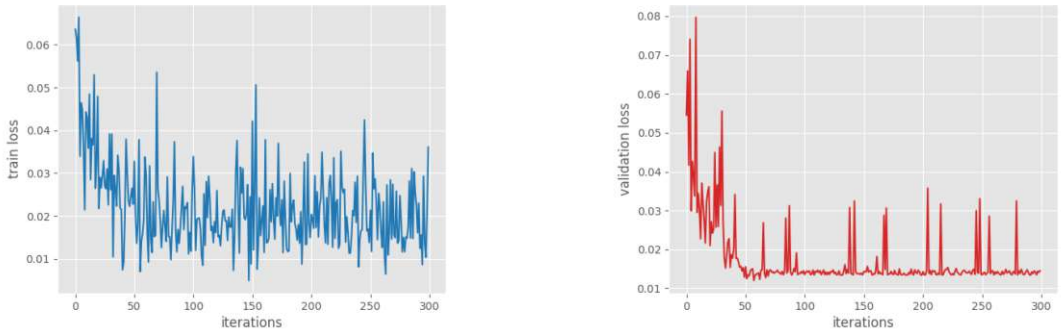
Based on the original F1Tenth object detection dataset in YOLO format, we created an instance of the F1Tenth object detection dataset in PASCAL VOC format as well as in COCO format for training Faster R-CNN and DETR. In order to train and evaluate both networks, we adapted the implementations of [85] and [86] for this work. Analogous to YOLOv7, Faster R-CNN and DETR have been trained and evaluated on the same platform for a total of 300 epochs. For Faster R-CNN, we used a batch size of four and a total of eight workers. On the other hand, a batch size of two and a total of four workers have been configured for training DETR. While Faster R-CNN has been trained from scratch, we finetuned DETR using the pretrained weights from the ResNet50 backboned DETR model trained on the COCO dataset. For evaluating DETR, transfer learning was necessary due to the long training time and slow convergence of the transformer on small datasets such as the F1Tenth object detection dataset compared to Microsoft COCO which has 330k images and 1.5 million object instances.

We applied the default hyperparameter configuration stated by the authors for training both networks. For training Faster R-CNN, we employ SGD as optimizer with an initial learning rate of $1e-2$, a momentum of 0.937 and a weight decay of $5e-4$. AMP training is performed while we rely on a learning rate scheduler applying the reduce learning rate on plateau strategy with a patience of ten epochs and a factor of 0.1. For data augmentation, horizontal image flipping as well as colorspace augmentation are used. All images are resized into shape (640, 640, 3). The detection head of the Resnet50 backboned Faster R-CNN model has been replaced and customized to comply with the number of classes in the F1Tenth object detection dataset. However, as Faster R-CNN

expects class zero to represent the background class (no object instance), the actual number of classes used for training and evaluation of the model is three.

For training DETR, AdamW is utilized as optimization algorithm with an initial learning rate of $1e^{-4}$ in the transformer, $1e^{-5}$ in the backbone and a weight decay of $1e^{-4}$. DETR applies the step learning rate scheduling policy using a step size of 200 epochs and a decay factor of 0.1. Horizontal image flipping as well as image scaling and cropping are used for data augmentation. All images are rescaled to have shape (800, 800, 3) and are normalized on ImageNet respectively. A dropout of 0.1 is applied for training the transformer while DETR is trained with a gradient clipping max norm of 0.1. The transformer utilizes six encoder-decoder layers and a total of eight self-attention heads. Finally, we set the number of object queries in the decoder to 100 and equally to Faster R-CNN, we specified $num\_classes = 3$ to compensate for the no-object/background class used by DETR. The simultaneous sharp drop in loss and significant rise in validation mAP score at epoch 200 when training DETR occurs from an adaption of the learning rate by the learning rate scheduler at this iteration of model training.

While figures 5.1 and 5.2 illustrate the training and validation performance of Faster R-CNN and DETR across all epochs, table 5.1 provides detailed statistics about the training time (in hours) and the number of parameters as well as compares FPS and mAP@.5, mAP@.5:.95 scores of native YOLOv7, Faster R-CNN and DETR on the test set of the F1Tenth object detection dataset.



(a) Training history

(b) Validation history

Figure 5.1: Faster R-CNN training performance across all epochs

Model	Training time [h]	Parameters	FPS	mAP@.5	mAP@.5:.95
Native YOLOv7	23.827	37 201 950	77.51	0.999	0.964
Faster R-CNN	63.73	41 081 886	13.21	0.943	0.893
DETR	142.5	41 279 752	28.57	0.999	0.953

Table 5.1: Training statistics of Faster R-CNN and DETR compared to native YOLOv7

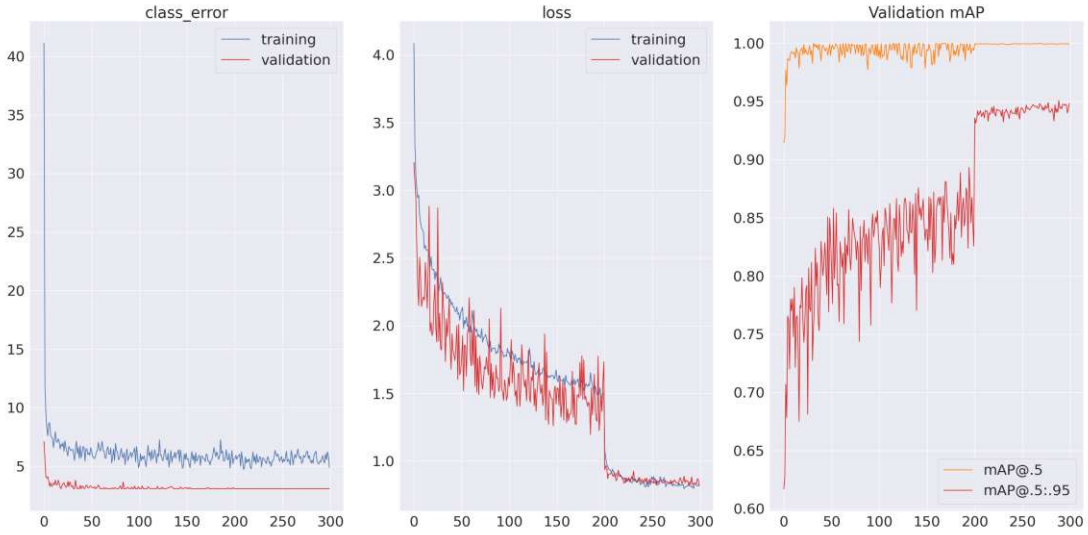


Figure 5.2: DETR training performance across all epochs

For a comprehensive evaluation of Att-YOLOv7 with the selected object detection baseline models on OOD data in Section 5.1.2, supplementary data from two different indoor track settings has been recorded. Equally to the F1Tenth object detection dataset, the data has been manually labelled and is specified in YOLO format. Thus, we further created an instance of both datasets in PASCAL VOC and COCO format for evaluating Faster R-CNN and DETR accordingly. One track was set up at Informatikhörsaal on 25^{th}, March 2023 while the other track was set up at Getreidemarkt 8/9 on 26^{th}, March 2023. For this reason, we denote the datasets as Informatikhörsaal dataset and Getreidemarkt 8/9 dataset respectively. Comparable to the initial recording of the F1Tenth object detection dataset, another F1Tenth car drove a couple of laps as well as boxes have been randomly placed on both tracks while recording. While figure 5.3 shows the track layouts of the tracks set up at Aufbaulabor on 12^{th}, April 2022 and Getreidemarkt 8/9 on 23^{rd}, August 2022, figure 5.4 illustrates the track layouts from which the Informatikhörsaal dataset and the Getreidemarkt 8/9 dataset have been sampled.

Regarding the environmental details, Aufbaulabor is a laboratory in the style of an old building with a hardwood floor. Contrarily, Getreidemarkt 8/9 is a seminar room in the style of a new building with a grey floor. Finally, Informatikhörsaal is a lecture hall with dark grey tiles. In order to mark the boarders of the tracks, we mostly used white ventilation hoses. We also want to highlight at this point that no duplicate track layouts were used across different rooms and recording dates such that the models are not biased towards a specific track setting. Hence, the supplementary data sampled at Getreidemarkt 8/9 vastly differs from the data originally recorded in this room i.e. the illumination, the surroundings and the track layout are mostly dissimilar.

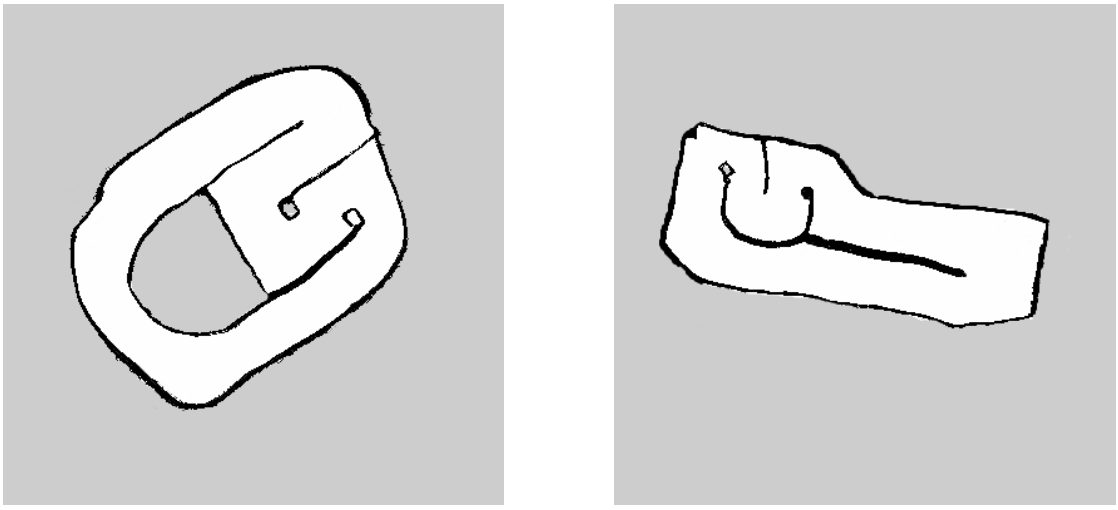


Figure 5.3: Track layouts composed at Aufbaulabor on 12^{th}, April 2022 (left) and Getreidemarkt 8/9 on 23^{rd}, August 2022 (right)

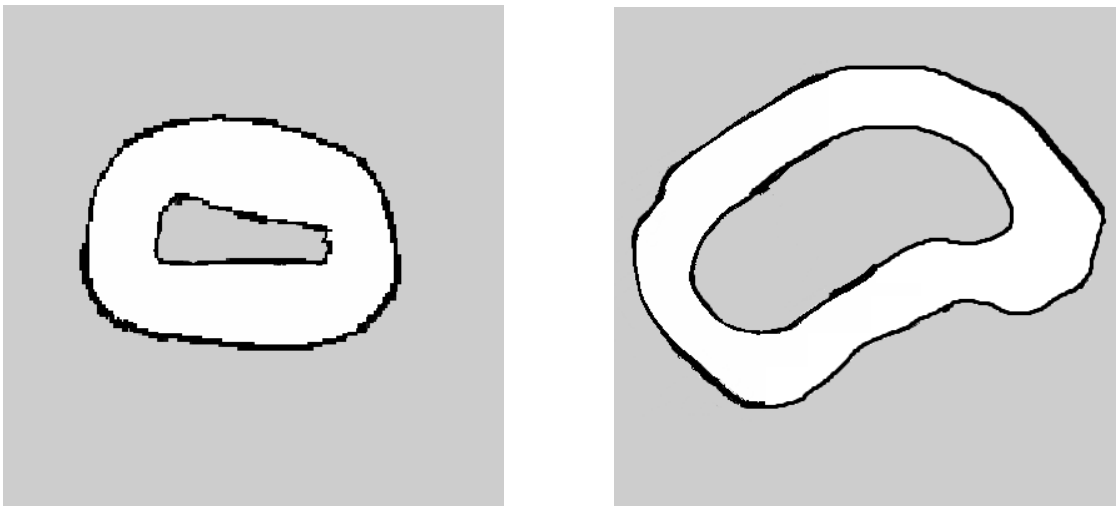


Figure 5.4: Track layouts composed at Informatikhörsaal on 25^{th}, March 2023 (left) and Getreidemarkt 8/9 on 26^{th}, March 2023 (right)

5.1.2 Evaluation on OOD data

For the purpose of performing an extensive analysis of our proposed system and the object detection baselines on illumination perturbed inputs, we varied the brightness of the test set of the F1Tenth object detection dataset, the Getreidemarkt 8/9 dataset and the Informatikhörsaal dataset in the range [25%, 200%] using a step size of 25%. Afterwards, all models have been successively evaluated on every brightness disturbance variation for each of the three datasets. In addition, we conducted an ablation study on the data augmentation techniques used for YOLOv7 focusing on the mosaic data

augmentation technique. To this end, we also trained native YOLOv7 and Att-YOLOv7 without mosaic data augmentation. These models are denoted as YOLOv7(-M) and Att-YOLOv7(-M) correspondingly.

First off, figure 5.5 visualizes a bar chart grouped by model architecture comparing mAP@.5:.95 of native YOLOv7, Att-YOLOv7, Faster R-CNN and DETR on the illumination perturbations of the test set of the F1Tenth object detection dataset. Complementary, figure 5.6 highlights mAP@.5:.95 of each object detection model as a function of the brightness niveau of the images in the evaluation dataset, showing the interpolated mAP@.5:.95 curve for each model. We used cubic splines to interpolate between the individually computed, discrete mAP@.5:.95 scores for this purpose. Given figures 5.5 and 5.6, native YOLOv7 and Att-YOLOv7 dominate in average mAP@.5:.95 score across all illumination variations. For a brightness level of 25%, DETR manages to outperform native YOLOv7 by a margin of 9.3%. Contrarily, Att-YOLOv7 slightly surpasses the results of DETR by 2.5%, achieving the highest mAP@.5:.95 score on all brightness variations. When comparing YOLOv7(-M) and Att-YOLOv7(-M), the gap in average mAP@.5:.95 score across all illumination variations jumps from 1.7% to 3% and is even larger than between YOLOv7 and Att-YOLOv7.

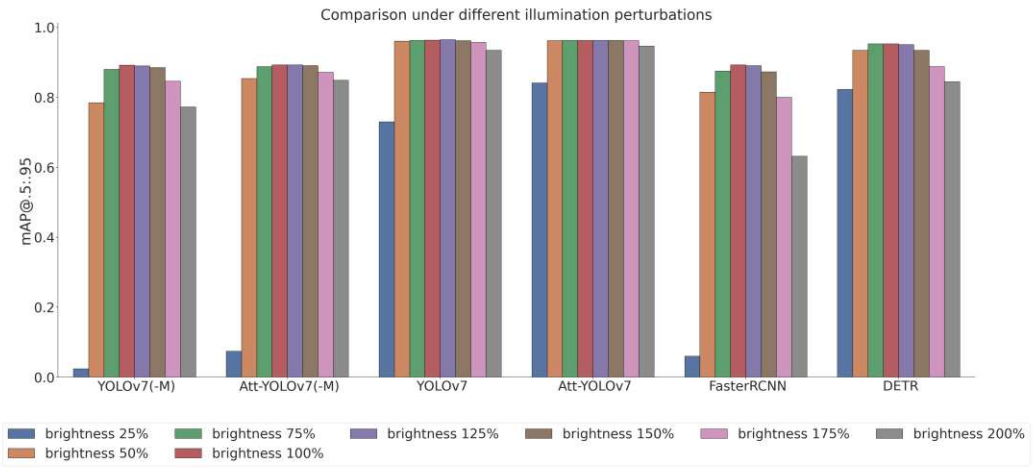


Figure 5.5: Evaluation of all object detection models on the test data of the F1Tenth object detection dataset

Equally to the evaluation of all object detection models on the test subset of the F1Tenth object detection dataset, figures 5.7 and 5.8 illustrate the trend in mAP@.5:.95 score of all models across all brightness variations on the Getreidemarkt 8/9 (GM89) dataset. Except for YOLOv7(-M) and Att-YOLOv7(-M), all object detection models achieve a similar mAP@.5:.95 score of approximately 72.6% on the non-illumination perturbed GM89 dataset. Considering all brightness variations, native YOLOv7 and Att-YOLOv7 clearly dominate in average mAP@.5:.95 score. While the gap in performance between native YOLOv7 and Att-YOLOv7 on the test data of the F1Tenth object detection dataset is

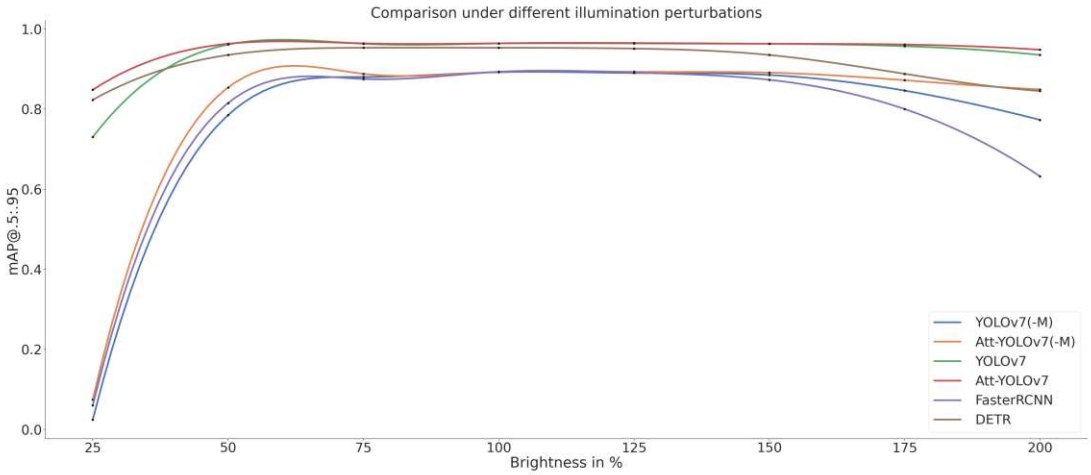


Figure 5.6: Illustration of the performance of all object detection models on the test data of the F1Tenth object detection dataset using cubic spline interpolation

only evident for a brightness level of 25%, Att-YOLOv7 outperforms native YOLOv7 on all illumination variations with a maximum gap of 8.9% in mAP@.5:.95 score for 175% and 200% brightness levels. For a brightness niveau of 25%, DETR achieves 14% and 5.7% higher mAP@.5:.95 scores than native YOLOv7 and Att-YOLOv7. In contrast, native YOLOv7 and Att-YOLOv7 outperform DETR on all other brightness perturbations with a peak offset of 15.1% and 23.4% in mAP@.5:.95 score for an illumination level of 200%. For YOLOv7(-M) and Att-YOLOv7(-M), we observe a slightly higher performance for YOLOv7(-M) on the brightness levels of 75% to 150% while Att-YOLOv7(-M) achieves better mAP@.5:.95 scores for brightness disturbance variations outside the range [75%, 150%].

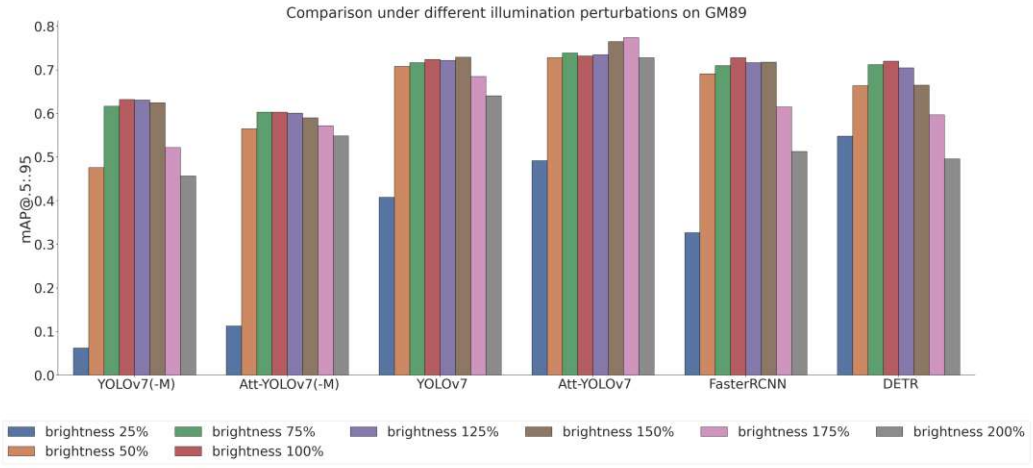


Figure 5.7: Evaluation of all object detection models on the Getreidemarkt 8/9 dataset

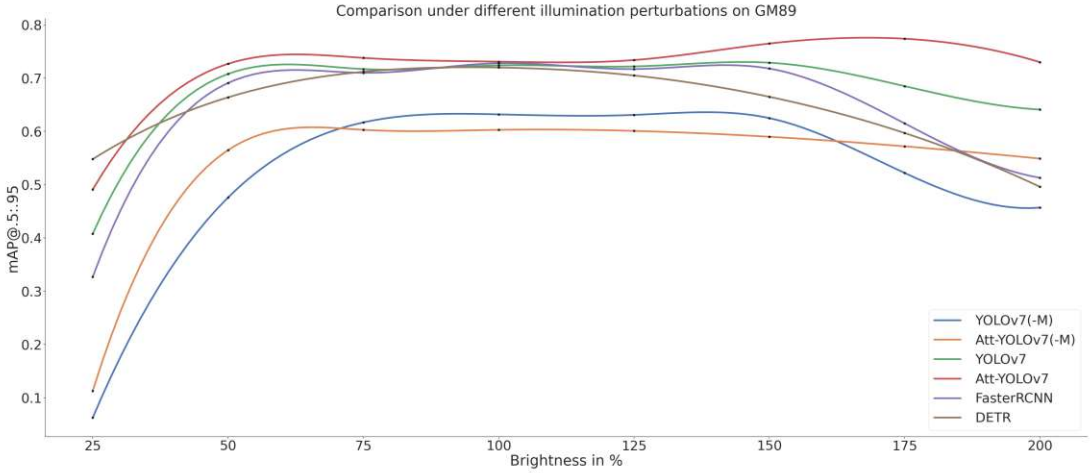


Figure 5.8: Illustration of the performance of all object detection models on the Getreidemarkt 8/9 dataset using cubic spline interpolation

Finally, figures 5.9 and 5.10 compare mAP@.5:.95 score of YOLOv7(-M), Att-YOLOv7(-M), native YOLOv7, Att-YOLOv7, Faster R-CNN and DETR on all brightness perturbations of the Informatikhörsaal dataset. While the gap in average mAP@.5:.95 score between native YOLOv7 and DETR across all illumination perturbations tends to close, Att-YOLOv7 dominates in average mAP@.5:.95 score across all brightness perturbations once more. In comparison to native YOLOv7, Faster R-CNN and DETR, Att-YOLOv7 achieves a 5.9%, 13.6% and 0.7% higher mAP@.5:.95 score. However, DETR clearly outperforms native YOLOv7 and Att-YOLOv7 by 38.73% and 30.8% in mAP@.5:.95 score for a brightness level of 25%. Considering our ablation study, we observe that

YOLOv7(-M) either matches or achieves a higher mAP@.5:.95 score compared to Att-YOLOv7(-M) for brightness perturbations greater equal to 100%. On the opposite, Att-YOLOv7(-M) reaches a higher mAP@.5:.95 score for 50% and 75% brightness levels.

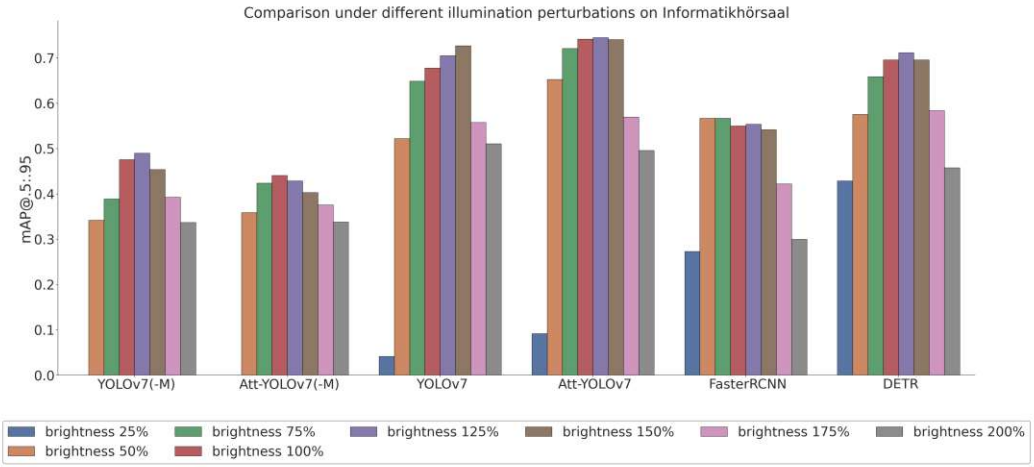


Figure 5.9: Evaluation of all object detection models on the Informatikhörsaal dataset

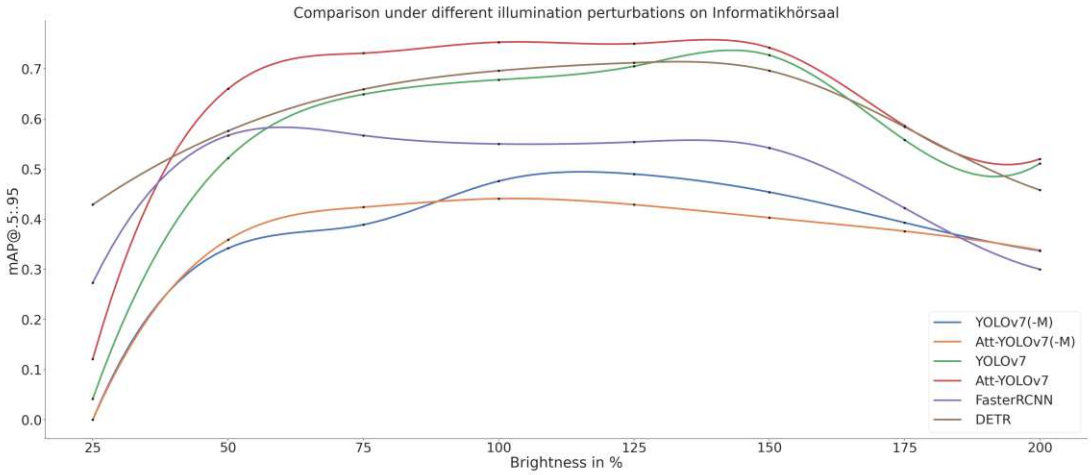


Figure 5.10: Illustration of the performance of all object detection models on the Informatikhörsaal dataset using cubic spline interpolation

Based on the experiments conducted in this section, we can conclude that Att-YOLOv7 demonstrates promising results on OOD data, specifically on illumination perturbed inputs compared to state-of-the-art object detection baseline models. Furthermore, our ablation study shows the tremendous impact of the mosaic data augmentation technique on the mAP@.5:.95 performance of the YOLOv7 algorithm as well as the dwindling performance gap between YOLOv7(-M) and Att-YOLOv7(-M). Our empirical

results indicate the benefit of combining human-attention feature maps and mosaic data augmentation to enhance and robustify the object detection accuracy of the YOLOv7 algorithm.

5.1.3 Comparison of Imitated Human Attention and Artificial Attention

Supplementary to comparing our approach with state-of-the-art object detection baselines on OOD data, we highlight the behavioral similarities and differences between the two different types of attention used in this work. Namely, the imitated human visual attention used by Att-YOLOv7 and the artificial attention also denoted as self-attention applied by DETR. In particular, we illustrate the imitated human-attention feature map implemented by Att-YOLOv7 and the decoder self-attention feature map utilized by the transformer of DETR. We explicitly compare the inference results gathered from Att-YOLOv7 and DETR on a sample image sequence of the test subset of the F1Tenth object detection dataset. In detail, we compare the inference results and the attention heatmaps of both models on the 25%, 100% (originally recorded data) and 200% brightness perturbed sample image sequence. Figure 5.11 presents the inference results of Att-YOLOv7 and the imitated human attention feature map depicted as overlayed orange heatmap for each frame of the three image sequences, where each column features one image sequence from top to bottom. Complementary, figure 5.12 adequately visualizes the inference results of DETR and the decoder self-attention feature map depicted as overlayed orange heatmap for each frame of the three image sequences.

While Att-YOLOv7 and DETR achieve approximately equivalent mAP.5:.95 scores on the 25% and 100% brightness perturbed test data, Att-YOLOv7 significantly outperforms DETR on the 200% brightness perturbed test data. When comparing the human-attention feature maps to the decoder self-attention feature maps, we observe that the human-attention feature maps produced by Att-YOLOv7 are more robust to the applied illumination perturbations. Furthermore, Att-YOLOv7 is also capable of detecting objects which receive no attentional focus in a given image. For instance, the fourth image of the middle column of figure 5.11 shows that the box is still detected, despite the human-attention being solely predicted on top of the F1Tenth racecar. Contrarily, DETR does not detect an object in a given image if the transformer does not attend to the object's location. Referring to the last two images of the right column of figure 5.12, we note that DETR does not manage to detect the box present in both images correctly, as the model did not attend to the specific locations in the images. For this reason, Att-YOLOv7 does not learn to fully rely on the imitated human-attention mechanism, but utilizes the auxiliary information provided by the human-attention feature map to enhance its object detection ability.

From a behavioural perspective, the decoder self-attention mechanism heavily focuses on the boundaries of objects present in an image. Conversely, the imitated human-attention mechanism is a time-dependent, sequential process, focusing on the near surroundings of an object, the boundaries of an object or the object itself. While there exists no complete

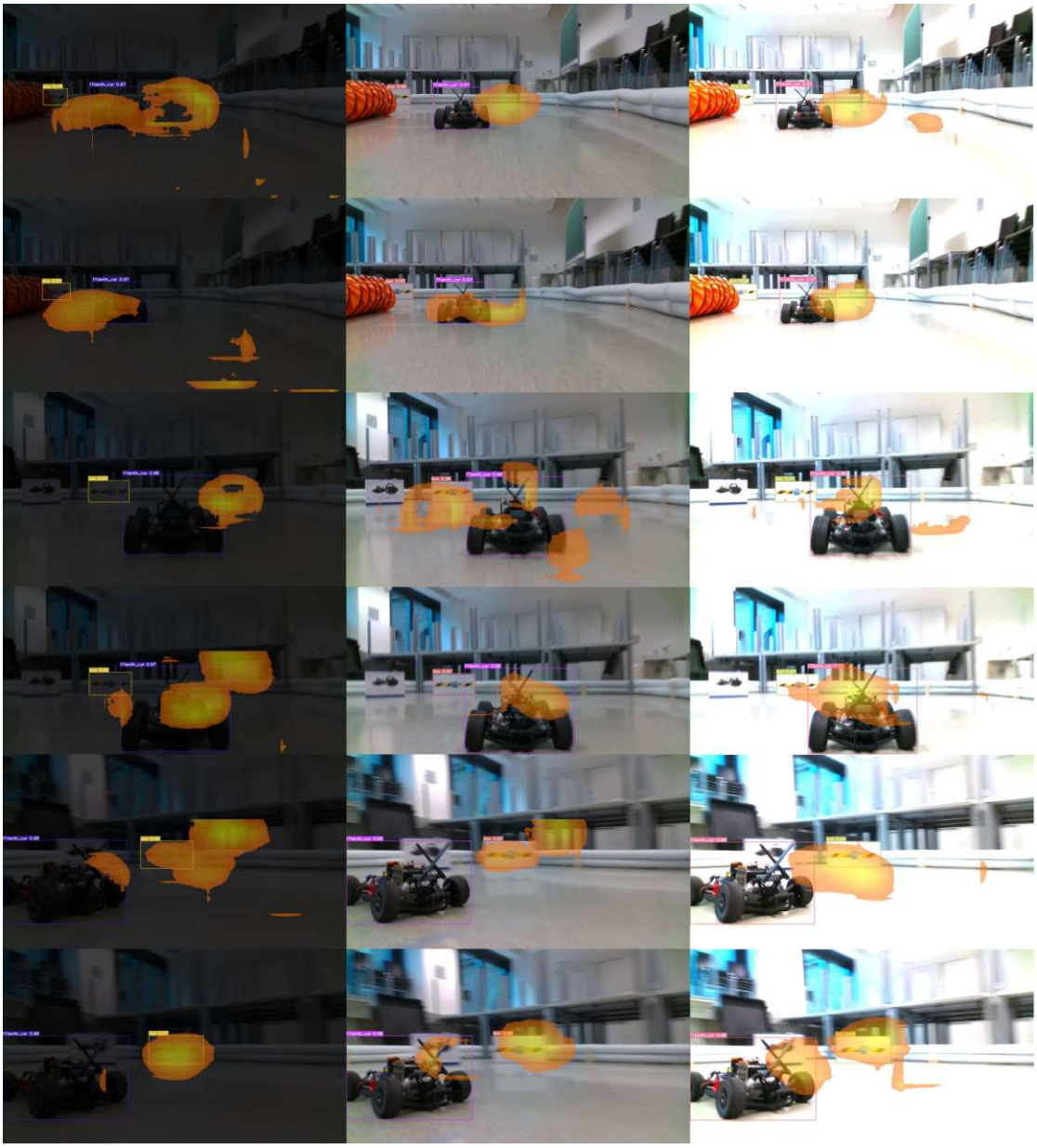


Figure 5.11: Att-YOLOv7 inference results of the 25%, 100% and 200% brightness perturbed image sequences with overlayed imitated human attention feature map

theory of human attention from a psychological or neurological point of view yet, our empirical results demonstrate the potential of utilizing human-attention for ML models, especially for object detection in the field of autonomous driving/racing.

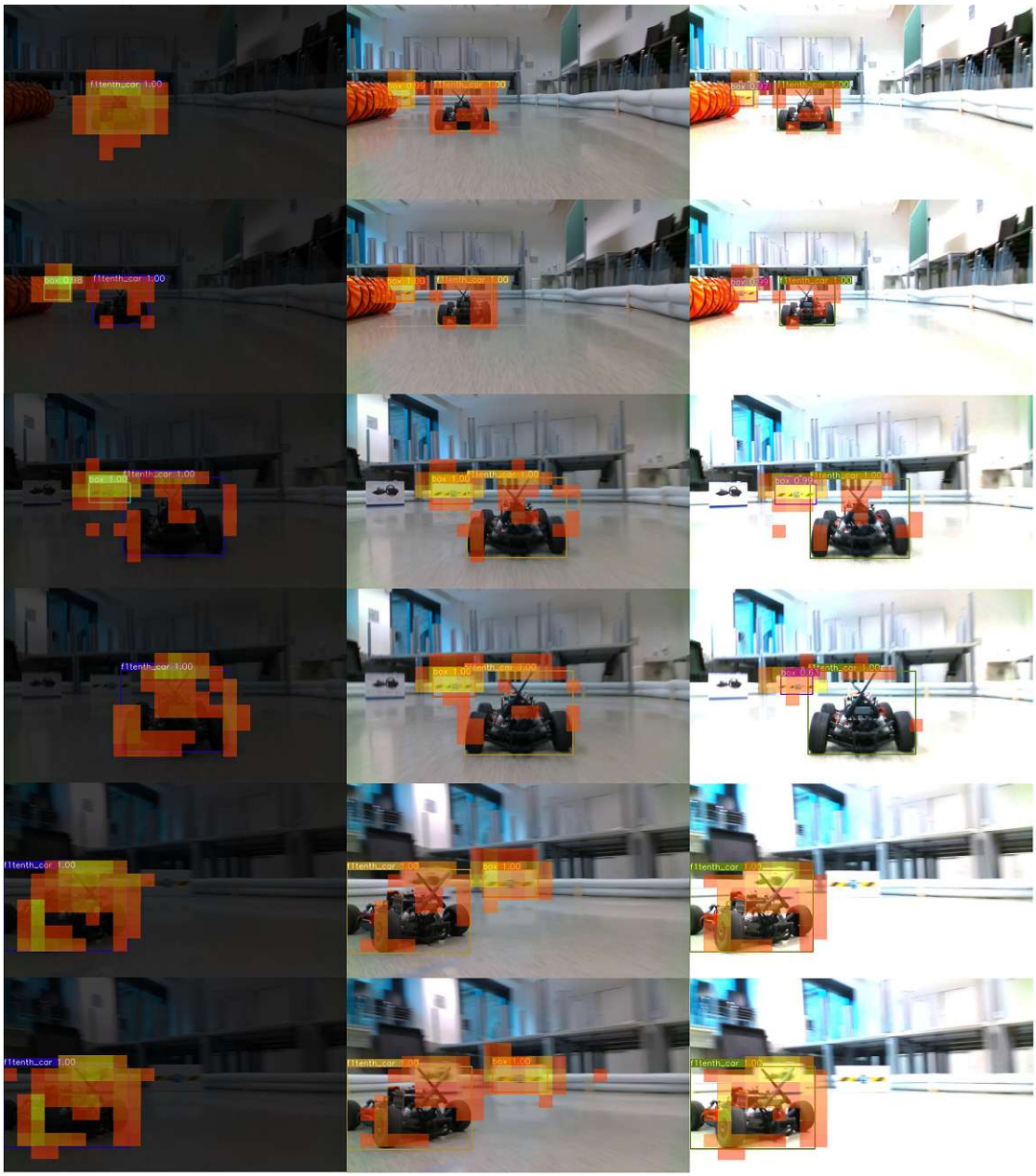


Figure 5.12: DETR inference results of the 25%, 100% and 200% brightness perturbed image sequences with overlayed decoder self-attention feature map

5.2 Limitations & AI-on-the-Edge

Concluding our exhaustive evaluation, we first demonstrate the limitations of our approach and afterwards show the feasibility of implementing Att-YOLOv7 for real-world autonomous driving/racing scenarios. We evaluated Att-YOLOv7 on the same platform all models were trained on. The test subset, composed of the images and labels located in the /images/test and labels/test directories of the F1Tenth object detection dataset, was used as the benchmark for this experiment. For a thorough comparison, native YOLOv7 has also been evaluated on the test subset of the F1Tenth object detection dataset. The validation metrics used in this context are FPS, precision (P), recall (R), mAP@.5 and mAP@.5:.95. Table 5.2 demonstrates the results each system obtained on the given metrics.

System Architecture	**FPS**	**P**	**R**	**mAP@.5**	**mAP@.5:.95**
Native YOLOv7	53.76	0.998	1	0.999	0.963
Att-YOLOv7	15.65	0.998	0.999	1	0.964

Table 5.2: Comparison of native YOLOv7 and Att-YOLOv7

When comparing mAP@.5 and mAP@.5:.95 scores of YOLOv7 and Att-YOLOv7, both algorithms achieve similarly high mAP@.5 and mAP@.5:.95 scores of 99.9% and 96.3% for native YOLOv7 as well as 100% and 96.4% for Att-YOLOv7 on the test set of the F1Tenth object detection dataset. The main limitation of Att-YOLOv7 is the reduction in FPS by around 70% compared to native YOLOv7 due to the additional inference time introduced by integrating the human-attention model. While Att-YOLOv7 may not be comparative to native YOLOv7 in terms of inference speed, we show that the integration of human-attention feature maps in the object detection algorithm implemented by YOLOv7 does not weaken its object detection accuracy on non-illumination perturbed input data. Conversely, Section 5.1.2 sheds light on the huge performance gain when employing human-attention feature maps for object detection models on OOD data.

Aside from the simulation evaluation, we aim to show the feasibility of our approach for real world autonomous racing applications by comparing the evaluation results of native YOLOv7 and Att-YOLOv7 on the test images of the F1Tenth object detection dataset gathered in simulation with the results achieved by equal and lower precision models on the same test data on hardware. Namely, we compare the performance of YOLOv7 and Att-YOLOv7 using FP32, FP16 and INT8 model variants. In the context of this experiment, we directly evaluated FPS, precision (P), recall (R), mAP@.5 and mAP@.5:.95 on the NVIDIA Jetson Xavier NX. The results are shown in table 5.3.

For evaluating Att-YOLOv7 using lower precision formats, the human-attention model as well as 4-channel YOLOv7 have been converted into the corresponding TensorRT engine format. For instance for running Att-YOLOv7 INT8 on the racecar, we subsequently performed the PTQ calibration procedure for both networks before benchmarking the

System Architecture	FPS	P	R	mAP@.5	mAP@.5:.95
Native YOLOv7 FP32	5.21	0.998	1	0.999	0.966
Att-YOLOv7 FP32	1.76	0.998	0.999	1	0.965
Native YOLOv7 FP16	15.20	0.998	1	0.999	0.966
Att-YOLOv7 FP16	2.23	0.998	0.999	1	0.965
Native YOLOv7 INT8	24.51	0.998	1	1	0.947
Att-YOLOv7 INT8	9.12	0.998	0.999	1	0.953

Table 5.3: Sim2Real comparison of native YOLOv7 and Att-YOLOv7

overall system. When referring to the simulation results presented in table 5.2, we observe equivalently high mAP@.5 and mAP@.5:.95 scores for all model variants of native YOLOv7 and Att-YOLOv7 by directly evaluating them on the racecar. Taking inference time into consideration, FP32 native YOLOv7 and Att-YOLOv7 achieve 5.21 FPS and 1.76 FPS on the NVIDIA Jetson Xavier NX accordingly. In contrast, an increase of 191.7% for native YOLOv7 and 26.7% for Att-YOLOv7 in FPS utilizing FP16 precision is recorded. Finally, a rise of 370.4% for native YOLOv7 and 436.5% for Att-YOLOv7 in FPS using INT8 calibrated weights is noted. Despite native YOLOv7 being 168.75% faster than Att-YOLOv7 when deploying both algorithms using INT8 precision, we argue that depending on the system requirements, these results show the feasibility of our approach for the integration of Att-YOLOv7 in real-world autonomous driving/racing applications.

CHAPTER 6

Discussion of Att-YOLOv7

In the final chapter, we debate on the results of our experiments conducted in Chapter 5 and discuss the design of our AttNN based dynamic object detection system. In Section 6.1, we elaborate the benefits and drawbacks of our proposed system compared to state-of-the-art object detection models from the related works 2.1 as well as potential improvements of our system architecture and the limitations of our hardware setup and software stack for recording the human-attention data. In Sections 6.2 and 6.3, we conclude our work by answering the research questions stated in Chapter 1 and provide a future outline for possible research directions. Finally, we provide insights on our initial system design in Chapter 8.

6.1 Discussion on the System Foundations

The results achieved by our Att-YOLOv7 object detector on the robustness experiments conducted on OOD data in Section 5.1.2 shed light on the potential benefit of incorporating human-attention in object detection models for autonomous driving/racing systems. While all selected state-of-the-art computer vision-based object detection baselines have shown to perform well on samples similar to the training data distribution, we clearly observe a mediocre to drastic decline in mAP@.5:.95 score for all object detection baselines on OOD data, in particular on illumination perturbed input data. However, our proposed approach enables much more robust object detection, outperforming all object detection baseline algorithms in mAP@.5:.95 score on all brightness variations, except for the 25% brightness case, of each of the three datasets used during our experiments.

Despite the rapid rise in robustness of our object detection algorithm when utilizing the biological mechanics of human-attention compared to state-of-the-art approaches, our system faces a two-fold major drawback. First off, the lack of end-to-end learning hampers the general usability of our system compared to native YOLOv7, Faster R-CNN and DETR as the human-attention model as well as 4-channel YOLOv7 must be trained

individually and successively. Secondly, the acquisition of a proper human-attention dataset given a specific use case may be a challenging task. For our proof-of-concept evaluation, we were limited to the human gaze recording capabilities of the VPS as well as to the quality of the VPS calibration on each test subject respectively. On the contrary, we were further limited to the design of our hardware/software stack for simulating the control of the F1Tenth racecar from the car's perspective. In order to simulate this behaviour, we had to post-process the recorded data for each run and map the data acquired in VPS space to the data recorded in the car's local camera space. However, due to the head movements of the test subjects while steering the F1Tenth racecar, the detection of the projected camera live stream using the ArUco marker system was not successful for all VPS images of all runs. For the same reason, focus points recorded in VPS space which are not located within the boundaries of the projected camera live stream are discarded either. Hence, if no valid focus points are present for a given VPS image, our post-processing pipeline does not compute a matching frame from the car's local camera space but skip the data entirely. To this end, each run in our human-attention dataset contains several gaps of varying size in the total series of frames. Those are especially present in turns on the tracks as these sections lead to the highest intensity in head movements for the test subjects. Finally, no driver performance monitoring was carried out. Therefore, the quality of the human-attention data recorded for each test subject was not assessed objectively, but assumed to be of equal quality. This assumption may not necessarily be true though, as the focus of each test subject on the provided camera live stream and the events on the track may not match.

For the purpose of providing a proof-of-concept solution in this work, we may propose several advancements to our system architecture. First off, our human-attention model runs inference only using sequences of RGB images. For the reason of learning strong latent feature representations, the human-attention model may learn on multi-modal inputs such as sequences of RGB images coupled with sequences of LiDaR scan data. However, in order to consider multi-modal inputs, sensor fusion must be performed in order to synchronize multiple data input streams. Alternatively, learning human-attention feature maps using sequences of 3D point clouds obtained by a depth camera may also improve the expressiveness and generalization of the human-attention model.

While the feasibility of our proposed approach for real-world autonomous driving/racing applications is shown in Chapter 5, our two-stage object detection algorithm is still computationally expensive and leaves a considerable memory footprint, due to two NNs being simultaneously executed. In order to further optimize the utilization of the constrained hardware resources of embedded computing platforms such as the NVIDIA Jetson Xavier NX, a one-stage object detection algorithm utilizing imitated human-attention may be designed. A one-stage attentional object detector may utilize the hardware resources of embedded devices much more efficiently as well as significantly reduce latency and increase throughput of the system.

6.2 Discussion of the results regarding the research questions

To that end, we want to provide an argument on the results of this thesis by answering the research questions that have been formulated in Chapter 1.

Research question zero (RQ0) aims towards the creation of a diverse and expressive human-attention labeled dataset in driving situations. Specifically, the objective of RQ0 is to provide an answer on how to record human-attention in driving situations and how to compose a dataset using human-attention labels in this context.

RQ0: How can we obtain a rich enough labeled dataset for human attention in driving situations?

Following the design and creation research strategy, as described in Chapter 3, Chapter 4 presents the methodological approach and the technical components required for recording human-attention labels and composing a proper human-attention labeled dataset for driving scenarios. In particular, Section 4.1.1 illustrates the procedure and discusses the hardware/software stack necessary for recording human-attention labels in our small-scale F1Tenth setup. Furthermore, we explain the structure and the content of the human-attention dataset. All of these insights contribute in answering RQ0.

Research question one (RQ1) aims towards the goal of finding a suitable NN algorithm capable of learning human-attention feature maps based on our human-attention dataset. In particular, RQ1 provides an argument on how to appropriately represent and learn human-attention labels for suitable NN architectures and which model fits best to reproduce adequate human-attention feature maps.

RQ1: How to design a neural network capable of imitating human attention for object detection in autonomous driving?

For answering RQ1, Chapter 4 shows how to train fitting NN algorithms on suitable human-attention label representations. In more detail, in Section 4.1.2, we introduce two strategies for training suitable NN architectures on our human-attention dataset and demonstrate two different human-attention label representations in order to investigate how to ease the process of learning the mechanics of human-attention for the models. Moreover, we perform an exhaustive analysis on various NN algorithms in order to objectively assess the selection of the human-attention model architecture. Based on the comprehensive analysis conducted in Section 4.1.2, we provide an empirical answer to RQ1.

Research question two (RQ2) aims towards the design of an optimal architecture for an attentional object detector, the suitability of object detection models in our context and how our system performs compared to state-of-the-art computer vision-based object detection algorithms in their application domain.

RQ2: What is the comparative performance of state-of-the-art object detection approaches, their suitability in our context, and the optimal overall architecture for an attentional object detector?

For answering RQ2, Chapter 4 presents an argument why our selected object detection base algorithm is appropriate for our purpose. Specifically, Section 4.2 provides details on architectural improvements and enhanced training routines of the object detection base model, showing it's supremacy compared to state-of-the-art object detectors. Chapter 4 further presents our approach for integrating the human-attention model and the object detection model, illustrating the overall algorithm of our attentional object detector. Finally, Section 5.1.2 visualizes the results of the comparison of our proposed system with selected state-of-the-art baselines on various OOD scenarios. To this end, we provide an empirical answer to RQ2 based on the results of the experiments conducted in Section 5.1.2.

In order to sum up the results of this thesis, the discussion of the presented work fulfils the aims of the thesis to create a rich enough human-attention labeled dataset, train a suitable NN algorithm to imitate human-attention feature maps, create a robust attentional object detector and evaluate the attentional object detector compared to state-of-the-art object detection baselines on OOD data.

6.3 Future research directions

Ultimately, we provide a suggestion for future research directions. While we conducted all experiments in this thesis on a small-scale F1Tenth setup, recording human-attention in a real-world autonomous driving/racing setup and training/evaluating our approach on publicly available object detection datasets for autonomous driving may confirm or even surpass the results in object detection robustness gathered with our experiments. Moreover, recording human-attention and composing a human-attention labeled dataset based on monitoring the human-attention of drivers in real-world driving situations may increase the quality of the human-attention dataset as opposed to our simulated setup. Thus, the trained human-attention model may be more efficient in producing expressive human-attention feature maps, leading to an even more robust attentional object detector. Apart from the evaluation of our approach on a real-world use case, the relationship between human-attention and the artificial attention mechanism utilized by the transformer architecture may be comprehensively studied. By understanding the relationship and the mechanics of human-attention and artificial attention, we may be able to learn a unified attention mechanism, exploit the benefits of both of them and create far more efficient ML models.

CHAPTER 7

Conclusion

Autonomous driving systems aim towards the goal of minimizing road casualties, enhancing the mobility of people with inabilities, reducing CO2-emissions and congestion while providing an array of further advantages. In order to satisfy the strong requirements with respect to safety and reliability for HAD agents, an encompassing and robust perception of the environment is obligatory. However, most state-of-the-art object detection algorithms sustain a significant loss in performance on OOD data. Based on this motivation, this thesis investigated the impact of human-attention on enhancing robot learning by designing a more robust object detector. In order to achieve this goal, we recorded data from human interaction in a driving task utilizing our developed data-collection pipeline and propose a novel approach to mimic human-attention in driving situations using a ML model. We enrich the visual input with human-attention feature maps and assess its impact on the object detection task, observing a significant improvement in robustness to OOD samples when exploiting imitated human-attention data.

The results and the knowledge obtained in this thesis highlight the potential of incorporating human-attention into ML pipelines to improve robot learning and subsequently answer three research questions to fulfil the scientific requirements for a master thesis. Summarized, our findings suggest that the robustness and efficiency of ML models may rapidly be increased by utilizing human-attention features, posing major implications on the application of human-attention for other ML tasks. Future research may consider replicating our findings in a larger-scale real-world autonomous driving setting as well as study the relation between human-attention and artificial attention in order to further exploit the power of both attention mechanisms, creating even more efficient ML models.

CHAPTER 8

Appendix

This chapter presents our initial approach for the design of the AttNN based dynamic object detection system where we originally pursued the goal towards reducing the latency and increasing the inference speed of object detection algorithms through human-attention feature maps. Based on our evaluation results in simulation, this approach was not competitive to state-of-the-art object detectors in terms of object detection accuracy and speed. For this purpose, we did not pursue this approach any further. However, for the sake of completeness, we discuss the algorithmic design of this approach and elaborate on the problems arisen.

System Design. In order to reduce the latency of modern object detectors, the idea of our approach - denoted as RoI-based dynamic object detection system - was to extract multiple RoI from the original input image at the locations of RoI in the predicted human-attention feature map. In a second step, the extracted RoI which are much smaller than the original image are simultaneously processed by the object detection NN in one forward pass and the results are transformed into the original image space.

In more detail, the RoI based dynamic object detection system utilizes the predicted human-attention heatmap by the AttNN to locate the RoI in the original image. In an intermediate processing step, the predicted human-attention heatmap is resized to the original spatial dimensions of the input image and all local maxima in the resized, non-normalized attention heatmap are computed. The pixel coordinates of each local maxima are considered as center locations for each RoI to be extracted from the original input image. For computing the local maxima in the predicted attention heatmap, we essentially utilize maximum filtering and set the minimum distance between local maxima corresponding to the radius of the normally distributed gaussian covariances. Additionally, a minimum pixel intensity of 1 (0.4% of the maximum pixel intensity) was configured for finding local maxima according to the defined minimum peak intensity of the gaussian covariances when training the AttNN and precomputing the multi-frame aggregated human-attention heatmap labels. The actual RoI size is a hyperparameter

which defines the spatial resolution of the input of the YOLOv7 model used in the RoI based dynamic object detection system. Afterwards, we perform batched inference on the RoI using RoI-YOLOv7 which expects a normalized FP32 tensor of shape (N, 3, M, M) where N corresponds to the batch size and M corresponds to the RoI size. Then, the predicted 2D bounding box coordinates are transformed into the original input space w.r.t. the location of the RoI in the original image. For the transformation of the predicted 2D bounding box coordinates in the final step, no perspective transformation is required as the 2D bounding box coordinates for each RoI as well as the coordinates of each RoI in the original image are known.

Modifications to the F1Tenth object detection dataset. While the F1Tenth object detection dataset described in Section 4.2.1 is composed of 2D RGB images with a resolution of 848x480 pixels based on the used camera systems, the F1Tenth object detection dataset used for training and validation of RoI-YOLOv7 - referred to as RoI-F1Tenth object detection dataset - needs to be from a lower spatial resolution. By design of the RoI based dynamic object detection system, the input shape of the object detection NN equals the size of the RoI. However, the images and bounding box coordinates of the labels may not be resized directly as the object detection NN processes the RoI which is cropped from the full resolution image at the location of the predicted region of human attention by the AttNN. For this purpose, each image in the original F1Tenth object detection dataset is cropped at the center location of each bounding box in the original image w.r.t. the target shape of the RoI. The bounding box coordinates are transformed using a perspective transformation between the original image space and the RoI space where out-of-bounds coordinates are clipped w.r.t. the RoI shape. Ultimately, if center locations of two or more bounding boxes have less distance to each other than the RoI size, the RoI is cropped at the mean center location of all bounding boxes in the original image. Therefore, multiple object instances within one input image are also covered by the RoI-F1Tenth object detection dataset.

System Evaluation. In order to assess the impact of the RoI size on the performance of RoI-YOLOv7, we provide the results of the system by applying multiple RoI sizes. Specifically, we evaluated the system utilizing RoI sizes 128×128 and 256×256. According to native YOLOv7 and Att-YOLOv7, we adequately trained and validated RoI-YOLOv7 on the training and validation set of the corresponding RoI-F1Tenth object detection dataset instance. Figure 8.1 illustrates mAP@.5 and mAP@.5:.95 scores of native YOLOv7 and 4-channel YOLOv7 along both RoI-YOLOv7 variants on the training and validation set of the corresponding F1Tenth object detection dataset instance across all epochs. Moreover, table 8.1 compares training time (in hours), parametric complexity, precision (P), recall (R), mAP@.5 and mAP@.5:.95 scores of all stated models on the corresponding validation set.

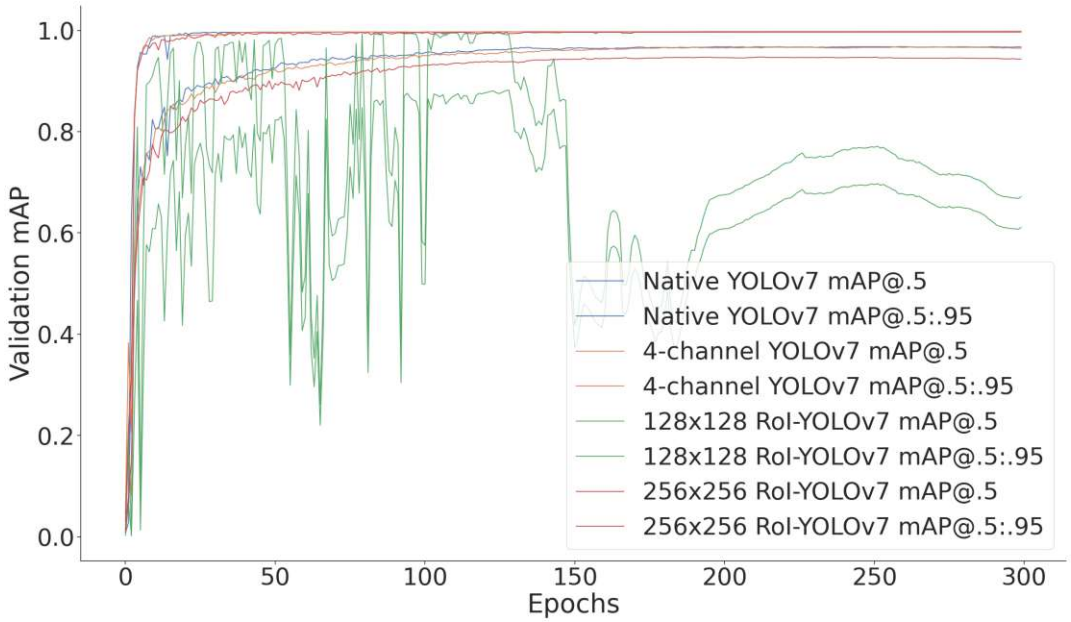


Figure 8.1: mAP scores of each YOLOv7 instance on the validation set

Model	**Training time [h]**	**Parameters**	**P**	**R**	**mAP@.5**	**mAP@.5:.95**
Native YOLOv7	23.827	37201950	0.998	0.998	0.998	0.967
4-channel YOLOv7	25.92	37202238	0.997	1	1	0.963
128 × 128 RoI-YOLOv7	4.713	37201950	0.638	0.968	0.672	0.611
256 × 256 RoI-YOLOv7	6.953	37201950	0.998	0.992	0.997	0.943

Table 8.1: Characteristic comparison of YOLOv7 variants

Equivalent to native YOLOv7 and Att-YOLOv7, we evaluated both instances of the RoI-based dynamic object detection system on the test set of the F1Tenth object detection dataset. Once more, we provide a comprehensive comparison of native YOLOv7, Att-YOLOv7 and both RoI-based dynamic object detection system instances on relevant metrics such as FPS, precision (P), recall (R), mAP@.5 and mAP@.5:.95. Table 8.2 demonstrates the results each system obtained on the given metrics. For simplicity, both instances of the RoI-based dynamic object detection system are denoted as RoI-YOLOv7.

System Architecture	FPS	P	R	mAP@.5	mAP@.5:.95
Native YOLOv7	53.76	0.998	1	0.999	0.963
Att-YOLOv7	15.65	0.998	0.999	1	0.964
128×128 RoI-YOLOv7	3.11	0.317	0.0719	0.047	0.0183
256×256 RoI-YOLOv7	3.20	0.708	0.403	0.338	0.183

Table 8.2: Comparison of native YOLOv7, Att-YOLOv7 and both RoI-YOLOv7 instances

When comparing mAP@.5 and mAP@.5:.95 scores of all systems, an immense decline for both RoI-YOLOv7 configurations in contrast to native YOLOv7 is recorded. In particular, 128×128 RoI-YOLOv7 merely achieves 4.7% and 1.83% in mAP@.5 and mAP@.5:.95 while 256×256 RoI-YOLOv7 only scores 33.8% and 18.3% in mAP@.5 and mAP@.5:.95 respectively. In comparison to the scores of the given metrics on the validation set of the F1Tenth object detection dataset for the RoI based YOLOv7 models, a drop of 62.5% and 59.27% for 128×128 RoI-YOLOv7 as well as a descent of 65.9% and 76% for 256×256 RoI-YOLOv7 is noted. We argue that this rapid decrease in performance results from the non-optimal RoI extraction centered at each local maxima of the predicted human-attention feature maps as well as from the lack of long-term attention memorization. Generally, objects from the original input space may be occluded in the RoI due to varying shifts between an object's location in the image and the position of the local maxima in the human-attention feature map which results in a lower IoU score and further a lower mAP score when transforming the prediction back to the original input space. Moreover, the RoI extraction is entirely dependent on the predicted attention heatmap. However, the predicted human-attention regions dynamically change across successive frames. Thus, objects that have been attended to at time step t might not constantly be attended to at time step $t + 1$ resulting in the same object not being detected in the succeeding frame.

List of Figures

Acronyms

AdaGrad Adaptive Gradient Descent. 18

Adam Adaptive Moment Estimation. 18, 19

AdamW Adam with decoupled weight decay. 48, 67

AI Artificial Intelligence. 1, 2, 6, 9, 23, 24, 37

AMP Automatic Mixed Precision. 48, 59, 66

API Application Programming Interface. 37, 63

AttNN Attentional Neural Network. 36–39, 45, 46, 49, 51, 53–55, 59, 61–66, 79, 85, 86, 89

BCE Binary Cross Entropy. 48, 58

BoF Bag-of-Freebies. 34, 56

BoS Bag-of-Specials. 34

BPTT Backpropagation Through Time. 14

CIoU Complete Intersection over Union. 34, 58

CNN Convolutional Neural Network. 6, 7, 11, 12, 24, 25, 27, 30, 31, 34

COCO Common Objects in Context. 7, 55, 66, 68

ConvLSTM Convolutional LSTM. 15, 49–52

CSPNet Cross Stage Partial Network. 33

cuDNN CUDA Deep Neural Network. 37

DDS Data Distribution Service. 38

DETR Detection Transformer. 2, 34, 36, 55, 65–68, 70–72, 74, 76, 79, 89, 90

DIoU Distance Intersection over Union. 34

DNN Deep Neural Network. 6, 7, 10–13, 16, 21, 24, 27, 37, 56

DSC Dice Similarity Coefficient. 27, 32, 48–53

E-ELAN Extended Efficient Layer Aggregation Networks. 56

ELAN Efficient Layer Aggregation Networks. 56

ELU Exponential Linear Unit. 13

EMA Exponential Moving Average. 47, 48, 57, 58

FBL Frame based Learning. 45–47, 49, 52

FCNN Fully Connected Neural Network. 30, 31, 33

FFNN Feed-Forward Neural Network. 14, 19, 34, 89

FoV Field of View. 1, 36

FP16 16-bit floating point precision. 48–50, 63, 64, 77, 78

FP32 32-bit floating point precision. 48–50, 57, 63, 64, 77, 78, 86

FPGA Field Programmable Gate Array. 41

FPN Feature Pyramid Network. 33, 58, 59

FPS Frames per Second. 40, 41, 67, 77, 78, 87, 88

GPU Graphical Processing Unit. 20, 37, 48, 49, 60, 63

HAD Highly Automated Driving. 36, 54, 61–64, 83

IL Imitation Learning. 16

INT8 8-bit integer. 64, 77, 78

IoU Intersection over Union. 27, 32, 33, 57, 58, 60, 88

JSON JavaScript Object Notation. 66

LiDaR Light Detection and Ranging. 23

LIP lateral intraparietal. 8

LSTM Long-Short Term Memory. 14, 15, 46, 50, 89, 91

mAP Mean Average Precision. 36, 55, 56, 58, 60–62, 64, 65, 67, 70–74, 77–79, 86–90

MDP Markov Decision Process. 16

ML Machine Learning. xiii, 5, 6, 9, 10, 16, 17, 21, 23, 24, 26, 28, 45, 54, 75, 82, 83

MLP Multilayer Perceptron. 11, 12

MSE Mean Squared Error. 49

NAG Nesterov Accelerated Gradient. 18

NaN Not a Number. 13

NLP Natural Language Processing. 5, 7, 14

NMS Non Maximum Suppression. 31, 33, 34, 57, 58

NN Neural Network. 6, 10, 12–14, 16, 17, 19, 20, 25–27, 36, 37, 39, 45, 46, 49, 50, 53–56, 65, 80–82, 85, 86

ONNX Open Neural Network Exchange. 37, 49, 63

OOD Out-of-Distribution. xiii, 1, 63, 65, 68, 69, 73, 74, 77, 79, 82, 83

OpenCV Open Source Computer Vision Library. 42, 43

OTA Optimal Transport Assignment. 58

PAN Path Aggregation Network. 33, 49, 51, 52

PASCAL VOC Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes. 7, 55, 66, 68

PCA Principal Component Analysis. 17

PTQ Post-Training Quantization. 64, 77

R-CNN Regions with CNN features. 6, 7, 30, 31, 55, 65–68, 70, 72, 79, 89

Radar Radio Detection and Ranging. 23

ReLU Rectified Linear Unit. 12, 13, 29

RL Reinforcement Learning. 16

RMS-Prop Root Mean Square Propagation. 18

RNN Recurrent Neural Network. 14, 15, 46, 89

RoI Regions of Interest. 7–9, 30, 36, 45, 59, 85–88

ROS Robot Operating System. 36–38, 40, 63, 64

RPN Region Proposal Network. 30

SBL Sequence based Learning. 45–47, 49, 50

scSE Concurrent Spatial and Channel Squeeze & Excitation. 49–53

SGD Stochastic Gradient Descent. 18, 58, 66

SPP Spatial Pyramid Pooling. 33

SSD Single Shot Detector. 6, 31–33, 55

SVM Support Vector Machine. 17, 30

TL Transfer Learning. 16

uint8 8-bit unsigned integer. 47, 63

VPS View Point System. 40–43, 80

VRAM Video Random Access Memory. 49, 60

XML Extensible Markup Language. 66

YOLO You Only Look Once. 6, 32–34, 47, 55–75, 77–79, 86–90

YOLOR You Only Learn One Representation. 57

Bibliography

[1] F. Foundation, "F1tenth." Website: https://f1tenth.org/, 2016 - 2022. Online; accessed 11 May 2022.

[2] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, and H. Lu, "Attention couplenet: Fully convolutional attention coupling network for object detection," *IEEE Transactions on Image Processing*, vol. 28, no. 1, pp. 113–126, 2019.

[3] K. Hara, M.-Y. Liu, O. Tuzel, and A. massoud Farahmand, "Attentional network for visual object detection," 2017.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *CoRR*, vol. abs/2005.12872, 2020.

[5] A. Johnson and R. W. Proctor, *Attention: Theory and practice*. Sage, 2004.

[6] J. Tsotsos, "Analyzing vision at the complexity level," *Behavioral and Brain Sciences*, vol. 13, 09 1990.

[7] J. Tsotsos and A. Rothenstein, "Computational models of visual attention," *Scholarpedia*, vol. 6, p. 6201, 01 2011.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[9] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[10] E. Sood, S. Tannert, D. Frassinelli, A. Bulling, and N. T. Vu, "Interpreting attention models with human visual attention in machine reading comprehension," in *Proceedings of the 24th Conference on Computational Natural Language Learning*, (Online), pp. 12–25, Association for Computational Linguistics, Nov. 2020.

[11] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[12] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto, "The surprising effectiveness of representation learning for visual imitation," *CoRR*, vol. abs/2112.01511, 2021.

[13] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," *CoRR*, vol. abs/1901.09005, 2019.

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[15] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," *CoRR*, vol. abs/1812.05069, 2018.

[16] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016. RoLoD: Robust Local Descriptors for Computer Vision 2014.

[17] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1691–1703, PMLR, 13–18 Jul 2020.

[18] D. Moyer, S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan, "Evading the adversary in invariant representation," *CoRR*, vol. abs/1805.09458, 2018.

[19] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020.

[21] Z. Fang, J. Wang, L. Wang, L. Zhang, Y. Yang, and Z. Liu, "SEED: self-supervised distillation for visual representation," *CoRR*, vol. abs/2101.04731, 2021.

[22] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," vol. 3, pp. 458–488.

[23] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object detection and tracking, based on dnn, for autonomous vehicles: A review," *IEEE Sensors Journal*, vol. 21, pp. 5668–5677, 2021.

[24] R. Nabati and H. Qi, "Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles," *ArXiv*, vol. abs/2009.08428, 2020.

[25] K. Qian, S. Zhu, X. Zhang, and E. L. Li, "Robust multimodal vehicle detection in foggy weather using complementary lidar and radar signals," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 444–453, 2021.

[26] A. Balasubramaniam and S. Pasricha, "Object detection in autonomous vehicles: Status and open challenges," *CoRR*, vol. abs/2201.07706, 2022.

[27] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, "YOLO-Z: improving small object detection in yolov5 for autonomous vehicles," *CoRR*, vol. abs/2112.11798, 2021.

[28] R. G. Joseph Redmon, Santosh Divvala and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[29] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[30] D. E. Christian Szegedy, Alexander Toshev, "Deep neural networks for object detection," 2013.

[31] Q. Lu, C. Liu, Z. Jiang, A. Men, and B. Yang, "G-cnn: Object detection via grid convolutional neural network," *IEEE Access*, vol. 5, pp. 24023–24031, 2017.

[32] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[33] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[34] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[35] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, K. Michael, J. Fang, imyhxy, Lorna, C. Wong, Z. Yifu, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, tkianai, yxNONG, P. Skalski, A. Hogan, M. Strobel, M. Jain, L. Mammana, and xylieong, "ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations," Aug. 2022.

[36] C. Wang, I. Yeh, and H. M. Liao, "You only learn one representation: Unified network for multiple tasks," *CoRR*, vol. abs/2105.04206, 2021.

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, pp. 21–37, Springer International Publishing, 2016.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.

[39] R. Girshick, "Fast r-cnn," 2015.

[40] K. H. Shaoqing Ren and J. S. Ross Girshick, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[41] Y. L. Jifeng Dai and J. S. Kaiming He, "R-fcn: Object detection via region-based fully convolutional networks," 2016.

[42] P. D. Tsung-Yi Lin, K. H. Ross Girshick, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016.

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[44] Q. Lai, S. Khan, Y. Nie, H. Sun, J. Shen, and L. Shao, "Understanding more about human and machine attention in deep neural networks," *IEEE Transactions on Multimedia*, vol. PP, pp. 1–1, 07 2020.

[45] DataDrivenInvestor, "Deep learning explained in 7 steps – updated." Website: https://www.datadriveninvestor.com/deep-learning-explained/, 2022. Online; accessed 06 March 2023.

[46] A. Eliasy and J. Przychodzen, "The role of ai in capital structure to enhance corporate funding strategies," *Array*, vol. 6, p. 100017, 07 2020.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] R. T. J. J., "Lstms explained: A complete, technically accurate, conceptual guide with keras." Website: https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2, 2020. Online; accessed 20 August 2023.

[49] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *CoRR*, vol. abs/1506.04214, 2015.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[51] A. Oppermann, "Underfitting and overfitting in deep learning." Website: https://medium.com/mlearning-ai/underfitting-and-overfitting-in-deep-learning-687b1b7eb738, 2021. Online; accessed 06 March 2023.

[52] S. Karagiannakos, "Localization and object detection with deep learning." Website: https://towardsdatascience.com/localization-and-object-detection-with-deep-learning- 67b5aca67f22, 2019. Online; accessed 06 March 2023.

[53] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[54] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," *CoRR*, vol. abs/1807.10165, 2018.

[55] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018.

[56] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," 2019.

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*, pp. 346–361, Springer International Publishing, 2014.

[58] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," 2018.

[59] B. J. Oates, *Researching Information Systems and Computing*. SAGE Publications, Ltd., 2012 ed., 2005.

[60] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decision Support Systems*, vol. 15, no. 4, pp. 251–266, 1995.

[61] P. Checkland, "Soft Systems Methodology: A Thirty Year Retrospective," *Systems Research and Behavioral Science*, vol. 17, pp. S11–S58, 2000.

[62] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, pp. 75–105, 3 2004.

[63] J. Hughes and T. Wood-Harper, "Systems development as a research act," *Journal of Information Technology*, vol. 14, no. 1, pp. 83–94, 1999.

[64] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system."

[65] S. C. Adam Paszke, Sam Gross and G. Chanan, "Pytorch." Website: https://pytorch.org/, 2016 - 2022. Online; accessed 15 July 2022.

[66] "Onnx." Website: https://onnx.ai/, 2019 - 2022. Online; accessed 15 July 2022.

[67] N. Corporation, "Cuda." Website: https://developer.nvidia.com/cuda-zone, 2007 - 2022. Online; accessed 15 July 2022.

[68] N. Corporation, "Tensorrt." Website: https://developer.nvidia.com/tensorrt, 2007 - 2022. Online; accessed 29 September 2022.

[69] V. GmbH, "Viewpointsystem." Website: https://viewpointsystem.com/, 2016 - 2022. Online; accessed 11 May 2022.

[70] freedesktop.org, "Gstreamer." Website: https://gstreamer.freedesktop.org/, 2008 - 2022. Online; accessed 04 July 2022.

[71] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, 2018.

[72] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016.

[73] L. OpenShot Studios, "Openshot." Website: https://www.openshot.org/de/, 2008 - 2022. Online; accessed 04 July 2022.

[74] S. U. Felix Resch, Daniel Scheuchenstuhl, "Eye-tracking." Website: https://gitlab.tuwien.ac.at/cyber-physical-systems/lehre/thesis/attention/eye-tracking, 2022. Online; accessed 04 July 2022.

[75] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[76] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[77] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017.

[78] M. Lechner, R. Hasani, P. Neubauer, S. Neubauer, and D. Rus, "Pyhopper – hyperparameter optimization," 2022.

[79] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," *CoRR*, vol. abs/1812.01187, 2018.

[80] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018.

[81] P. Iakubovskii, "Segmentation models pytorch." Website: https://github.com/qubvel/segmentation_models.pytorch, 2019. Online; accessed 09 May 2023.

[82] A. Canziani, "pytorch-cortexnet." Website: https://github.com/Atcold/pytorch-CortexNet, 2018. Online; accessed 05 March 2023.

[83] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[84] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[85] S. R. Rath, "A simple pipeline to train pytorch faster rcnn object detection model." Website: https://debuggercafe.com/a-simple-pipeline-to-train-pytorch-faster-rcnn-object-detection-model/, 2021. Online; accessed 09 May 2023.

[86] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "De:tr: End-to-end object detection with transformers." Website: https://github.com/facebookresearch/detr, 2020. Online; accessed 09 May 2023.