

Entwicklung eines Fahr-Copilots -  
Ein multimodaler Deep-Learning-Ansatz für präventive Echtzeit-  
Bremseignis-Vorhersage in kritischen Fahrsituationen

**Studienarbeit**

des Studienganges Informatik  
an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Florian Kulig

Juni 2025

Bearbeitungszeitraum	40 Wochen
Matrikelnummer, Kurs	5384281, INF22D
Dualer Partner	Capgemini, Leinfelden-Echterdingen

# Abstract

## Deutsch

Advanced Driver Assistance Systems (ADAS) durchlaufen eine technologische Evolution von regelbasierten Pipeline-Architekturen zu End-to-End-Neural-Networks. Diese Arbeit entwickelt eine multimodale Machine-Learning-Architektur zur präventiven Vorhersage von Bremseignissen auf Edge-Computing-Hardware unter realistischen Automotive-Bedingungen.

Die implementierte Lösung kombiniert YOLO-basierte Objekterkennung mit proprietären OBD-II-Fahrzeugdaten durch systematisches Reverse Engineering. Mittels Mode-22-Command-Exploration wurde erstmals direkter Zugriff auf Audi-spezifische Bremssignale (PID 0x3F9F) etabliert. Die entwickelte modulare Drei-Komponenten-Architektur (Input Encoder, Fusion Module, Output Decoder) ermöglicht die systematische Evaluation verschiedener Verarbeitungsstrategien für multimodale Sequenzverarbeitung.

Der aufgenommene Datensatz umfasst 42.686 Sequenzen aus zwölf Autobahnfahrten mit einer charakteristischen Klassenimbalance von 1:36 für Bremseignisse. Die experimentelle Evaluation demonstriert die Überlegenheit Transformer-basierter Sequenzmodellierung gegenüber LSTM-Architekturen bei extremer Klassenimbalance. Das finale Modell erreicht eine ROC Area Under Curve von 0,973 für die Bremseignis-Vorhersage bei einer True Positive Rate von 92,2% und einer True Negative Rate von 96,8%.

Die Implementierung auf einem Raspberry Pi 5 (8GB RAM) demonstriert die Echtzeitfähigkeit mit einer End-to-End-Pipeline-Latenz von 132,75 ms unter den definierten Hardware-Constraints. Die systematische Architektur-Evaluation etabliert eine reproduzierbare Methodik für multimodale Fahrerassistenzsysteme und trägt zur methodischen Fundierung automobiler KI-Systeme bei.

# Abstract

## English

Advanced Driver Assistance Systems (ADAS) are undergoing a technological evolution from rule-based pipeline architectures to end-to-end neural networks. This work proposes a multimodal machine learning architecture for the proactive prediction of braking events on edge computing hardware under realistic automotive conditions.

The implemented solution integrates YOLO-based object detection with proprietary OBD-II vehicle data through systematic reverse engineering. The utilization of Mode 22 command exploration facilitated the establishment of direct access to Audi-specific brake signals (PID 0x3F9F) for the first time. The developed modular three-component architecture (input encoder, fusion module, output decoder) enables the systematic evaluation of different processing strategies for multimodal sequence processing.

The recorded real-world dataset encompasses 42,686 sequences obtained from twelve highway trips, exhibiting a characteristic class imbalance of 1:36 for braking events. The experimental evaluation demonstrates the superiority of transformer-based sequence modeling over LSTM architectures in cases of extreme class imbalance. The final model demonstrates an area under the curve (AUC) of 0.973 for the purpose of predicting braking events, exhibiting a true positive rate of 92.2% and a true negative rate of 96.8%.

The implementation on a Raspberry Pi 5 (8GB RAM) demonstrates real-time capability with an end-to-end pipeline latency of 132.75 milliseconds under the defined hardware constraints. The systematic architecture evaluation establishes a reproducible methodology for multimodal driver assistance systems and contributes to the methodological foundation of automotive AI systems.

# Inhaltsverzeichnis

Eigenständigkeitserklärung .....	II
Abstract .....	III
Inhaltsverzeichnis .....	V
Abkürzungsverzeichnis .....	VIII
Abbildungsverzeichnis .....	IX
1. Einleitung .....	1
1.1 Motivation .....	1
1.2 Problemstellung .....	1
1.3 Zielsetzung .....	2
2. Stand der Technik .....	3
2.1 ADAS und autonome Fahrsysteme .....	3
2.2 Deep Learning für Fahrzeugwahrnehmung .....	6
2.2.1 Kamerabasierte Objekterkennung .....	6
2.2.2 Attention-Mechanismen in der Umgebungswahrnehmung .....	6
2.2.3 Temporale Modellierung: LSTM vs. Transformer für Sequenzdaten .....	7
2.2.4 Multimodale Daten: Early vs. Late vs. Intermediate Fusion .....	8
2.3 Hardwarebeschränkungen in der Automotive-Domäne .....	9
3. Systemdesign und Architektur .....	10
3.1 Problemformalisierung .....	10
3.2 Systemspezifikation und Design-Entscheidungen .....	12
4. Datengrundlage und -preprocessing .....	14
4.1 OBD-II Reverse Engineering .....	14
4.1.1 Grundlagen der OBD-II Kommunikation .....	14
4.1.2 Proprietäre Diagnosemodi und Mode 22 .....	14

4.1.3 Differentielle Analyse zur Bremsignal-Identifikation .....	15
4.1.4 Binäre Signaldekodierung und Protokollanalyse.....	16
4.1.5 Validierung und Korrelationsanalyse.....	16
4.1.6 Sicherheitsaspekte und Risikominimierung .....	17
4.2 YOLO-Finetuning für das Autobahnszenario.....	18
4.2.1 Empirische Baseline-Evaluation und Problemidentifikation .....	18
4.2.2 Datensatz-Analyse und Auswahlmethodik .....	18
4.2.3 Implementierung und Optimierungsresultate .....	19
4.3 Datenerfassung und -verarbeitung .....	20
4.4 Multimodales Feature Engineering.....	22
4.5 Datensatzcharakteristiken .....	25
5. Modellarchitektur und Implementierung.....	27
5.1 Gesamtarchitektur und Datenfluss .....	27
5.1.1 Dreistufige Verarbeitungsarchitektur.....	27
5.1.2 Temporale Sequenzverarbeitung .....	28
5.1.3 End-to-End-Optimierung trotz Modularität .....	28
5.2 Input Encoder Implementierung .....	29
5.3 Fusion Module Implementierung .....	30
5.4 Output Decoder Architekturen .....	31
5.4.1 Gemeinsame Multi-Task Klassifikationsstruktur .....	31
5.4.2 LSTMOutputDecoder .....	31
5.4.3 TransformerOutputDecoder .....	32
5.4.4 Architektur-Vergleich .....	32
6. Experimentelle Evaluation .....	33
6.1 Experimenteller Aufbau und Methodik.....	33
6.2 Loss-Funktion Evolution und Task-Balancing .....	33

6.2.1 Weighted Binary Cross-Entropy .....	33
6.2.2 Focal Loss Implementation .....	35
6.3 Architektur-Vergleich: LSTM vs. Transformer Decoder.....	36
6.3.1 Performance-Analyse.....	36
6.3.2 Qualitative Decoder-Charakteristika .....	37
6.4 Transformer Attention Muster.....	39
6.4.1 Gelernte zeitliche Abhangigkeiten.....	39
6.4.2 Temporale Sequenzmodellierung .....	39
6.5 Edge Computing Performance .....	40
6.6 Einfluss von Sequenzuberlappung .....	44
7. Diskussion.....	45
7.1 Methodische Erkenntnisse .....	45
7.1.1 Modularer Architektur-Ansatz.....	45
7.1.2 Sequenzmodellierung und Hardwarelimitierungen .....	45
7.2 Technische Limitationen.....	46
7.2.1 Umgebungsmodellierung .....	46
7.2.2 Datensatz .....	47
7.2.3 Hardwarelimitierungen .....	47
8. Fazit und Ausblick.....	48
8.1 Wissenschaftliche Beitrage .....	48
8.2 Ausblick .....	49
Literaturverzeichnis.....	51

# Abkürzungsverzeichnis

ADAS .....	Advanced Driver Assistance System
AI .....	Artificial Intelligence
ARM .....	Advanced RISC Machine
CAN .....	Controller Area Network
CNN .....	Convolutional Neural Network
COCO .....	Common Objects in Context (Datensatz)
CPU .....	Central Processing Unit
FP .....	Floating Point
FPS .....	Frames per Second
FSD .....	Full Self Driving
Hz .....	Hertz
ISO .....	International Organization for Standardization
LiDAR .....	Light Detection and Ranging
LSTM .....	Long short-term memory
MPC .....	Model Predictive Control
MPS .....	Metal Performance Shaders
NPU .....	Neural Processing Unit
OBD .....	On Board Diagnostics
PID .....	Parameter Identification Number (OBD-II)
PR-AUC .....	Precision-Recall Area Under Curve
RAM .....	Random Access Memory
ROC-AUC .....	Area Under the Receiver Operating Characteristic Curve
RPN .....	Region Proposal Network
RRT .....	Rapidly-exploring Random Trees
TOPS .....	Trillions of Operations Per Second
VLM .....	Vision Language Model
VW .....	Volkswagen
YOLO .....	You Only Look Once

# Abbildungsverzeichnis

Abbildung 1: Verkehrssituation mit markiertem relevantem Bildbereich.....	13
Abbildung 2: Abfrage-Code für OBD-II Mode 22 Addressraum.....	15
Abbildung 3: Code für Dekodierung des Bremssignals .....	16
Abbildung 4: Fahrzeugtelemetrie mit Bremsphasen .....	17
Abbildung 5: Performance bei Bildkompression .....	21
Abbildung 6: PR- und ROC-Kurven bei Bremsvorhersage .....	34
Abbildung 7: PR- und ROC-Kurven bei Rollvorhersage .....	34
Abbildung 8: Precision-Recall-Kurve bei Brems- und Rollvorhersage.....	36
Abbildung 9: ROC-Kurve bei Brems- und Rollvorhersage.....	37
Abbildung 10: Normalisierte Verwechslungsmatrizen für Brems- und Rollvorhersagen.....	38
Abbildung 11: Recency Bias für Sequenzmodellierung bei Transformer.....	39
Abbildung 12: Attention-Muster für Sequenzmodellierung bei Transformer .....	39
Abbildung 13: Leistungs- und Speicherverbrauchsmessung des multimodalen Modells .....	41
Abbildung 14: Performanceverbesserung der Modellarchitekturen nach Optimierungen .....	42

# 1. Einleitung

## 1.1 Motivation

Advanced Driver Assistance Systems (ADAS) durchliefen eine technologische Evolution von regelbasierten Pipeline-Architekturen zu End-to-End-Neural-Networks, exemplarisch dargestellt durch Teslas Full Self-Driving (FSD) Paradigma [1]. Deutsche Autobahnen präsentieren spezifische Herausforderungen durch unbegrenzte Geschwindigkeiten ( $>130$  km/h), hohe Verkehrsdichte und komplexe Spurwechselmanöver. Kritische Fahrsituationen manifestieren sich primär als Stauende-Szenarien, Auffahrsituationen und dynamische Überholmanöver externer Verkehrsteilnehmer [2], [3].

Statistische Analysen deutscher Verkehrsunfälle zeigen erhöhte Unfallschwere bei Autobahngeschwindigkeiten, wobei Auffahrunfälle 20% aller schweren Verkehrsunfälle repräsentieren [3]. Insgesamt stieg die Anzahl der Unfälle auf deutschen Autobahnen um jährlich mehr als 4,5% [4]. Die durchschnittliche menschliche Reaktionszeit für Notbremsungen beträgt 0,7-1,3 Sekunden, bestehend aus Wahrnehmungszeit (0,5s - 0,7s) und motorischer Reaktionszeit (0,2s - 0,3s) [5], [6], [7].

Präventive Fahrerassistenzsysteme operieren durch Früherkennung kritischer Situationen, wodurch Zeitreserven für Fahrerintervention geschaffen werden. Der in dieser Arbeit entwickelte multimodale Deep-Learning-Ansatz integriert dafür YOLO-basierte Objekterkennung mit proprietären OBD-II-Fahrzeugdaten durch systematisches Reverse Engineering.

## 1.2 Problemstellung

Präventive Bremsereignis-Vorhersage erfordert die zeitkritische Klassifikation 1-5 Sekunden vor Eintreten des Ereignisses. Diese Zeitspanne überschreitet die menschliche Reaktionszeit und ermöglicht präventive Systeminterventionen im Gegensatz zu reaktiven Notremssystemen.

Die technische Implementierung konfrontiert drei primäre Herausforderungen: Erstens erfordert multimodale Datenfusion die Synchronisation heterogener Datenquellen (Kamera, OBD-II) bei gleichzeitiger Feature-Extraktion für binäre Klassifikation

zukünftiger Fahrzeugverzögerung. Zweitens limitieren Hardware-Constraints des Raspberry Pi 5 8GB (ARM Cortex-A76) die Modellkomplexität bei Latenz-Anforderungen <200ms, wie es für Notbremssysteme typisch ist [8]. Drittens erschweren proprietäre OBD-II-Parameter die Extraktion kritischer Bremssignale ohne herstellerspezifische Reverse Engineering-Prozesse.

### 1.3 Zielsetzung

Die Zielsetzung dieser Arbeit ist die Entwicklung und Evaluation einer multimodalen Deep-Learning-Architektur zur präventiven Bremseignis-Vorhersage unter realistischen Edge-Computing-Bedingungen. Im Fokus steht dabei die Etablierung einer systematischen Methodik für die kontrollierte Bewertung multimodaler Fusionsstrategien in sicherheitskritischen Automotive-Anwendungen.

Die technischen Ziele umfassen die Entwicklung einer modularen Deep-Learning-Architektur, die eine isolierte Analyse einzelner Systemkomponenten ermöglicht. Durch systematisches OBD-II Reverse Engineering soll erstmals direkter Zugriff auf proprietäre Fahrzeugelektrometrie etabliert und in die multimodale Datenverarbeitung integriert werden.

Die praktische Implementierung adressiert die Echtzeitfähigkeit auf dem Raspberry Pi 5 als repräsentative Automotive-Edge-Computing-Plattform. Performance-Metriken quantifizieren dabei die Trade-offs zwischen Klassifikationsleistung und Hardware-Constraints (Latenz, Speicherverbrauch) unter realen Fahrbedingungen.

Die wissenschaftliche Zielsetzung liegt in der Etablierung einer reproduzierbaren Evaluationsmethodik für multimodale Fahrerassistenzsysteme, die eine evidenzbasierte Architekturentscheidung durch kontrollierte Komponentenvariation ermöglicht. Dies schafft eine methodische Grundlage für weiterführende Forschungsarbeiten in der automobilen KI-Domäne.

## 2. Stand der Technik

### 2.1 ADAS und autonome Fahrsysteme

Die Entwicklung von Fahrerassistenzsystemen und autonomen Fahrzeugen basiert auf drei fundamentalen Architekturansätzen, die sich in ihrer algorithmischen Herangehensweise und Implementierungsstrategie grundlegend unterscheiden. Diese Ansätze repräsentieren verschiedene Paradigmen der maschinellen Wahrnehmung und Entscheidungsfindung in komplexen Verkehrsumgebungen.

#### Traditionelle regelbasierte Systeme

Traditionelle ADAS-Implementierungen folgen einer modularen Pipeline-Architektur, die sequenzielle Verarbeitungsschritte durch explizit definierte Regelsysteme realisiert. Diese Architektur gliedert sich typischerweise in die Komponenten Wahrnehmung (Perception), Prädiktion (Prediction), Pfadplanung (Path Planning) und Kontrolle (Control). Jede Komponente operiert als eigenständiges Modul mit klar definierten Eingabe- und Ausgabeschnittstellen. [9], [10], [11]

Die Wahrnehmungskomponente transformiert Sensordaten von Kameras, LiDAR-Systemen und Radarsensoren in strukturierte Umgebungsrepräsentationen. Objekterkennung erfolgt durch klassische Computer-Vision-Algorithmen wie Support Vector Machines oder frühe Convolutional Neural Networks mit nachgelagerten Klassifikationsstufen [12], [13]. Die Prädiktion zukünftiger Objekttrajektorien basiert auf kinematischen Modellen und regelbasierten Verhaltensannahmen, beispielsweise konstanter Geschwindigkeit oder Beschleunigung [14], [15].

Pfadplanungsalgorithmen wie A\* oder Rapidly-exploring Random Trees (RRT) generieren kollisionsfreie Trajektorien unter Berücksichtigung expliziter Verkehrsregeln und Sicherheitsabstände. Die Kontrollebene implementiert diese Trajektorien durch PID-Regler oder Model Predictive Control (MPC) für Längs- und Querregelung des Fahrzeugs.

Der zentrale Vorteil regelbasierter Systeme liegt in ihrer Interpretierbarkeit und Vorhersagbarkeit. Entscheidungsprozesse sind durch explizite Regeln nachvollziehbar, was Validierung und Zertifizierung erleichtert. Gleichzeitig begrenzt die starre

Regelstruktur die Adoptionsfähigkeit an unvorhergesehene Verkehrssituationen, die nicht in den vordefinierten Regelsätzen abgebildet sind. [16], [17]

### **Tesla Full Self-Driving Paradigma**

Das Tesla Full Self-Driving (FSD)-System implementiert eine modulare End-to-End-Architektur, die die klassische Pipeline-Struktur beibehält, jedoch alle Module durch Deep-Learning-Komponenten ersetzt, die gemeinsam End-to-End trainiert werden. Tesla verwendet die Hauptblöcke „Perception“ und „Planning & Control“ weiterhin, ersetzt jedoch die algorithmischen Implementierungen durch tiefe neuronale Netzwerke.

Die Wahrnehmungskomponente basiert auf der HydraNet-Architektur. Dabei handelt es sich um ein Multi-Task-Learning-System mit einem gemeinsamen neuronalen Netzwerk-Backbone und aufgabenspezifischen Decoder-Zweigen für die Objekterkennung, die Verkehrslichterkennung und die Spurprädiktion. Die Eingabeschicht verarbeitet Bilder von acht Kameras mittels einer Bird's Eye View-Transformation durch Transformer-Module. [1] Tesla ergänzt die HydraNet-Architektur durch Occupancy Networks, die volumetrische Bird's-Eye-View-Repräsentationen und Occupancy Flow für das 3D-Szenenverständnis generieren. Diese Netzwerke prognostizieren für jeden Voxel<sup>1</sup> im 3D-Raum die Belegungswahrscheinlichkeit und Bewegungsrichtung und ermöglichen so die Differenzierung zwischen statischen und dynamischen Objekten. [18]

Die Planungskomponente eliminiert traditionelle Algorithmen zugunsten einer Deep-Learning-basierten Pfadplanung. Das neuronale Planungsmodul generiert Trajektorien durch Reinforcement-Learning-Techniken, die auf Millionen realer Fahrsituationen trainiert wurden.

Der Unterschied zu traditionellen Systemen liegt im End-to-End-Training: Alle Module werden durch gemeinsame Verlustfunktionen optimiert, während sie gleichzeitig modular feinabgestimmt werden können. Diese Architektur ermöglicht eine gradientenbasierte Optimierung des gesamten Systems bei gleichzeitiger Beibehaltung der modularen Interpretierbarkeit. [1], [9], [10], [11]

---

<sup>1</sup> Dreidimensionales Bildelement

## **Hybride Ansätze**

Hybrid-Architekturen kombinieren die Vorteile regelbasierter Interpretierbarkeit mit der Adaptivität neuronaler Netzwerke durch modulare Systeme mit differenzierbaren Komponenten. Diese Ansätze implementieren klassische Pipeline-Strukturen, ersetzen jedoch einzelne Module durch neuronale Netzwerke, die durch gemeinsame Verlustfunktionen optimiert werden.

Typische Implementierungen verwenden neuronale Netzwerke für Wahrnehmungsaufgaben wie Objekterkennung und Szenenverständnis, während Pfadplanung und Kontrolle durch differenzierbare Optimierungsverfahren realisiert werden [19]. Differenzierbare Pfadplaner wie Neural Motion Planning integrieren Gradienteninformationen aus nachgelagerten Modulen zur End-to-End Optimierung der gesamten Pipeline [20].

Die Architektur ermöglicht selektive Integration von Domänenwissen durch explizite Constraints in Optimierungsproblemen, beispielsweise Verkehrsregeln als harte Nebenbedingungen oder Komfortkriterien als Regularisierungsterme [21]. Attention-basierte Fusionsmechanismen kombinieren Ausgaben verschiedener Module unter Berücksichtigung ihrer jeweiligen Unsicherheiten. [22]

Hybrid-Systeme adressieren die Limitationen rein regelbasierter und End-to-End Ansätze durch selektive Anwendung neuronaler Komponenten bei Erhaltung interpretierbarer Entscheidungsstrukturen. Die erhöhte Systemkomplexität erfordert jedoch abgestimmte Trainingsstrategien und erweiterte Validierungsmethoden.

Die drei vorgestellten Paradigmen repräsentieren komplementäre Ansätze mit spezifischen Vor- und Nachteilen bezüglich Leistung, Interpretierbarkeit und Implementierungsaufwand. Die Wahl der geeigneten Architektur hängt von Anwendungsanforderungen, Sicherheitskriterien und regulatorischen Rahmenbedingungen ab.

## 2.2 Deep Learning für Fahrzeugwahrnehmung

Deep-Learning-Methoden haben die automatisierte Fahrzeugwahrnehmung durch effiziente Objekterkennung, temporale Sequenzmodellierung und multimodale Sensorfusion revolutioniert. Zu den Kernherausforderungen zählen die Echtzeit-Objektdetektion, die Modellierung zeitlicher Abhängigkeiten in Fahrzeugdaten und die Integration heterogener Sensormodalitäten. [23], [24], [25]

### 2.2.1 Kamerabasierte Objekterkennung

**Two-Stage Detectors** implementieren einen hierarchischen Detektionsansatz mit separaten Phasen für Region Proposal und Klassifikation. R-CNN nutzt Selective Search, um Kandidatenregionen zu erzeugen, und eine anschließende CNN-basierte Klassifikation. Fast R-CNN [26] optimiert diese Pipeline durch ROI-Pooling und End-to-End-Training, während Faster R-CNN das Region Proposal Network (RPN) integriert und eine vollständig lernbare Feature-Extraktion ermöglicht [27]. Diese Architekturen erreichen eine hohe Detektionsgenauigkeit durch die explizite Modellierung der Objektlokalisierung, müssen Daten jedoch sequenziell verarbeiten. [28], [29]

**One-Stage-Detektoren** eliminieren die Region-Proposal-Phase, indem sie Objekte in einem einzigen Forward-Pass direkt lokalisieren und klassifizieren. Die YOLO-Familie (You Only Look Once) unterteilt Eingabebilder in Gitterzellen und führt simultan Bounding-Box-Regression und Klassenklassifikation durch. YOLOv1 etablierte das Grundprinzip der Unified Detection, während nachfolgende Iterationen kontinuierliche Architekturverbesserungen integrierten. Diese Entwicklung zeigt eine systematische Reduktion der Inferenzlatenz bei verbesserter Detektionsleistung, wodurch sich One-Stage-Detektoren für latenzkritische Automotive-Anwendungen eignen. [29] Durch die Möglichkeit des domänen spezifischen Finetunings können Anpassungen an verkehrsspezifische Objektklassen und Szenarien vorgenommen werden. [30], [31], [32], [33]

### 2.2.2 Attention-Mechanismen in der Umgebungswahrnehmung

Attention-Mechanismen haben sich als Schlüsseltechnologie zur adaptiven Gewichtung relevanter Informationen in komplexen Verkehrsszenarien etabliert [22], [34]. **Self-Attention** ermöglicht eine dynamische Fokussierung auf wichtige Zeitpunkte oder

räumliche Regionen innerhalb einer Modalität. **Cross-Attention** erleichtert die Interaktion zwischen verschiedenen Datenströmen mittels Query-Key-Value-Mechanismen [35], [36].

**Multi-Head-Attention** erweitert diese Konzepte durch parallele Attention-Köpfe, die verschiedene Repräsentationsräume gleichzeitig erfassen. Dadurch ist die simultane Modellierung unterschiedlicher Aspekte wie räumlicher Beziehungen, temporaler Dynamiken und semantischer Korrespondenzen möglich.

**Spatial Attention** konzentriert sich auf relevante Bildbereiche und reduziert die Verarbeitungskomplexität durch selektive Feature-Extraktion.

### 2.2.3 Temporale Modellierung: LSTM vs. Transformer für Sequenzdaten

**LSTM-Netzwerke** [37] adressieren das Vanishing-Gradient-Problem [38], [39] rekurrenter Architekturen mithilfe von drei spezialisierten Gates: Input Gate, Forget Gate und Output Gate. Das Input Gate kontrolliert die Integration neuer Informationen, das Forget Gate selektiert die zu behaltenden Informationen aus dem Cell State und das Output Gate bestimmt die Aktivierung der Hidden States. Diese gating-basierte Architektur ermöglicht selektive Langzeitspeicherung und graduelles Vergessen irrelevanter Informationen. [40], [41]

Durch die sequenzielle Verarbeitung modellieren LSTMs temporale Abhängigkeiten mittels rekurrenter Verbindungen zwischen Zeitschritten. Bidirektionale LSTM-Varianten verarbeiten Sequenzen in beide Richtungen und kombinieren vergangene und zukünftige Kontextinformationen. Gestapelte LSTM-Architekturen ermöglichen eine hierarchische Feature-Extraktion über multiple Abstraktionsebenen. [37]

**Transformer-Architekturen** [42] revolutionierten die Sequenzmodellierung durch vollständig attention-basierte Mechanismen ohne rekurrente Verbindungen. Der Self-Attention-Mechanismus berechnet Gewichtungen zwischen allen Sequenzelementen parallel und ermöglicht so die direkte Modellierung langreichender Abhängigkeiten [43], [44]. Multi-Head-Attention erweitert dieses Konzept durch parallele Attention-Köpfe, die verschiedene Repräsentationsräume simultan erfassen. [44]

Bei der Scaled Dot-Product Attention werden Attention-Scores als Skalarprodukt zwischen Queries und Keys berechnet, die durch die Wurzel der Dimensionalität normalisiert werden. Positional Encoding kompensiert die fehlende sequenzielle Induktivität durch explizite Positionsinformationen. „Layer Normalization“ und „Residual Connections“ stabilisieren das Training tiefer Transformer-Architekturen. [42]

Ein architektonischer Vergleich zeigt fundamentale Unterschiede: LSTMs bieten eine inhärente sequenzielle Induktivität und geringere Speicheranforderungen durch rekurrente Parameter-Sharing [37]. Transformer ermöglichen eine Parallelisierung und eine direkte Modellierung von Langzeitabhängigkeiten [34], erfordern jedoch eine quadratische Speicherkomplexität bezüglich der Sequenzlänge [45]. Die kontinuierliche Entwicklung effizienter Transformer-Varianten wie Performer und Linformer adressiert diese Skalierungslimitationen [46], [47].

#### 2.2.4 Multimodale Daten: Early vs. Late vs. Intermediate Fusion

**Early Fusion** kombiniert Rohdaten oder Features niedriger Abstraktionsebenen in frühen Netzwerkschichten. Diese Strategie ermöglicht feingranuläre Modalitätsinteraktionen, erfordert jedoch synchronisierte Datenströme und ist anfällig für dominante Sensoren. [48], [49]

**Late Fusion** verarbeitet Modalitäten unabhängig bis zu hochdimensionalen Repräsentationen und kombiniert finale Entscheidungen oder Features. Vorteile umfassen Robustheit gegenüber Sensorausfällen und die Nutzung vortrainierter modalitätsspezifischer Modelle. [48]

**Intermediate Fusion** kombiniert Modalitäten auf mittleren Abstraktionsebenen und ermöglicht kontextuelle Verknüpfungen semantischer und sensor-spezifischer Informationen. Cross-Modal-Attention-Mechanismen bieten eine adaptive Gewichtung verschiedener Informationsströme, die sich an situative Anforderungen anpassen. [49]

Die Auswahl der Fusionsstrategie erfolgt anwendungsspezifisch unter Berücksichtigung von Latenz, Robustheit und verfügbaren Rechenressourcen. Hybride Ansätze kombinieren mehrere Fusionsebenen, um die komplementären Vorteile verschiedener Strategien zu nutzen. [49]

## **2.3 Hardwarebeschränkungen in der Automotive-Domäne**

Die Implementierung von Machine Learning-basierten ADAS-Systemen unterliegt strikten Hardware-Limitationen durch automotive-spezifische Anforderungen bezüglich Latenz, Energieverbrauch und Kostenbeschränkungen. Diese Limitierungen beeinflussen fundamentale Designentscheidungen in der Algorithmusentwicklung und bestimmen die Auswahl geeigneter Modellarchitekturen.

### **Edge Computing vs. Cloud Processing**

Für sicherheitskritische ADAS-Funktionen wie Notbremssysteme sind End-to-End-Latenzgarantien von unter 200 ms für die gesamte Verarbeitungskette erforderlich [50]. Aufgrund variabler Netzwerklatenzen schließen diese strikten Timing-Anforderungen eine cloudbasierte Verarbeitung kategorisch aus. Edge-Computing-Architekturen verlagern die ML-Inferenz auf fahrzeuginterne Hardware-Plattformen, um Netzwerkabhängigkeiten vollständig zu eliminieren.

Moderne Edge-Computing-Plattformen im Automobilbereich basieren auf ARM-basierten SoCs mit integrierten NPUs. Zusätzliche Kostenbeschränkungen durch Serienfertigung bei Millionen-Stückzahlen schließen High-End-GPU-Lösungen aus und favorisieren spezialisierte Edge-AI-Chips mit optimiertem Preis-Leistungs-Verhältnis. [51]

### **Hardware-optimierte Machine-Learning-Deployment-Strategien**

Zu den Optimierungsstrategien für eingebettete Machine-Learning-Systeme zählen die Modell-Quantisierung zur Reduzierung auf INT8-/INT16-/FP16-Präzision, die Knowledge Distillation zur Modellkompression sowie das strukturelle Pruning redundanter Netzwerkverbindungen. Diese Techniken reduzieren den Speicherbedarf um den Faktor 2-8 und den Rechenaufwand um den Faktor 2-18 bei erhaltener Genauigkeit. [51], [52]

Spezialisierte Inferenz-Frameworks wie OpenVINO, TensorRT oder ONNX Runtime ermöglichen hardwarespezifische Optimierungen durch Graphfusion und Anpassungen des Speicherlayouts. Die resultierende Latenzreduktion ist entscheidend, um die 200-ms-Anforderung bei gleichzeitiger Verarbeitung multimodaler Sensordatenströme einzuhalten. [53], [54], [55]

# 3. Systemdesign und Architektur

## 3.1 Problemformalisierung

Die zentrale Problemstellung dieser Arbeit besteht in der Entwicklung eines Machine-Learning-Systems zur frühzeitigen Vorhersage kritischer Fahrmanöver. Durch die Kombination visueller Verkehrssituationsdaten mit fahrzeuginternen Telemetrieinformationen soll das System präventive Fahrerassistenz ermöglichen.

### Mathematische Problemdefinition

Das Vorhersageproblem wird als multimodale Sequenz-zu-Binär-Klassifikation formalisiert. Eine Funktion

$$f: (X_{vision}, X_{telem}) \rightarrow \hat{y}$$

soll aus Trainingsdaten gelernt werden, die aus zeitlich synchronisierten Eingabedaten zukünftige Fahrhandlungen, wie „Bremsen“ oder „Rollen“ prognostiziert.

**Vision-basierte Verkehrssituation:**  $X_{vision} \in R^{T \times N \times D_{det}}$

wobei  $T$  die betrachtete Zeitsequenz,  $N$  die Anzahl der erkannten Verkehrsteilnehmer und  $D_{det}$  deren charakterisierende Merkmale (Position, Größe, Objekttyp) repräsentiert.

**Fahrzeuginterne Telemetrie:**  $X_{telem} \in R^{T \times D_{telem}}$

mit fahrzeugspezifischen Zustandsparametern wie Geschwindigkeit, Motorparametern und Fahrer-Inputs über die Zeitsequenz  $T$ .

**Prädiktionsaufgabe:**  $\widehat{y}_{t+\Delta t} \in \{0, 1\}$  für verschiedene Vorhersagehorizonte  $\Delta t$

Das System soll zwei komplementäre Fahrmanöver vorhersagen:

1. **Bremsereignisse:** Notwendigkeit aktiver Geschwindigkeitsreduktion
2. **Effizienzoptimierte Fahrweise:** Situationen zum energiesparenden "Ausrollen"

## Anwendungsdomäne und Systemziele

Die multimodale Vorhersage von Fahrmanövern adressiert zwei zentrale Anwendungsbereiche moderner Fahrerassistenzsysteme:

- **Sicherheitskritische Prävention:** Durch die frühzeitige Erkennung von Bremssituationen können präventive Sicherheitssysteme aktiviert werden. Eine Vorhersage mehrere Sekunden im Voraus ermöglicht die rechtzeitige Vorbereitung von Notbremsassistenten, Aufmerksamkeitswarnungen oder adaptiven Geschwindigkeitsregelungen. Dies trägt zur Reduktion von Auffahrunfällen und zur Erhöhung der Fahrsicherheit bei.
- **Effizienzoptimierung:** Die Vorhersage von Situationen, in denen eine Geschwindigkeitsreduktion ohne aktives Bremsen möglich ist („Rollen“, engl. „Coasting“), ermöglicht energieeffiziente Fahrstrategien. Durch das rechtzeitige Reduzieren der Motorleistung anstelle eines späteren Bremsens kann der Kraftstoffverbrauch signifikant gesenkt werden.

## Herausforderungen der multimodalen Fusion

Die Kombination von visuellen Verkehrssituationsdaten mit fahrzeuginternen Telemetrieinformationen ist mit spezifischen Herausforderungen verbunden:

- **Temporale Synchronisation:** Beide Datenmodalitäten müssen zeitlich exakt aufeinander abgestimmt werden, um kausale Zusammenhänge zwischen äußeren Verkehrssituationen und Fahrerreaktionen zu erfassen.
- **Skalendisparität:** Visuelle Objektdaten (diskrete Entitäten) und kontinuierliche Telemetriedaten haben eine unterschiedliche Dimensionalität und erfordern verschiedene Repräsentations- und Verarbeitungsansätze.
- **Echtzeitfähigkeit:** Die praktische Anwendung in Fahrzeugen erfordert Kompromisse bei der Modellkomplexität aufgrund begrenzter Rechenressourcen.

Die erfolgreiche Lösung dieser Problemstellung verspricht einen signifikanten Fortschritt in der Entwicklung präventiver, intelligenter Fahrerassistenzsysteme.

## **Systemanforderungen für Echtzeitfähigkeit**

Basierend auf den in Kapitel 2.3 analysierten Latenz-Constraints sicherheitskritischer ADAS-Systeme definiert sich die Anforderung für das entwickelte multimodale Verzögerungs-Vorhersagesystem. Während Notbremsassistenten eine maximale Latenz von 200ms tolerieren, wird in produktiven Deployments eine deutliche Unterschreitung dieser Obergrenze angestrebt, um Sicherheitsmargen zu gewährleisten und Raum für zusätzliche Verarbeitungsschritte in der kompletten ADAS-Pipeline zu schaffen. Das System muss daher eine Inferenz-Latency unter 200ms erreichen, wobei eine Minimierung der Verarbeitungszeit als kontinuierliches Optimierungsziel definiert wird.

## **Wissenschaftliche Einordnung**

Das entwickelte System implementiert eine modulare Drei-Komponenten-Architektur zur systematischen Evaluation multimodaler Fusionsstrategien. Die Trennung in Input Encoder, Fusion Module und Output Decoder ermöglicht die isolierte Analyse einzelner Architekturkomponenten und deren Beitrag zur Gesamtleistung. Die Gesamtarchitektur des Deep-Learning-Modells wird in Kapitel 5.1 detaillierter erläutert.

Die modulare Architekturphilosophie ermöglicht die systematische Identifikation optimaler Konfigurationen für spezifische Aufgabenstellungen durch kombinatorische Evaluation aller Komponentenvarianten. Diese Herangehensweise schafft eine empirisch vergleichbare Grundlage für evidenzbasierte Architekturentscheidungen in multimodalen Fahrerassistenzsystemen.

## **3.2 Systemspezifikation und Design-Entscheidungen**

Die praktische Umsetzung des multimodalen Systems basierte auf definierten Hardware- und Software-Spezifikationen. Diese gewährleisten die Reproduzierbarkeit der Experimente und die Übertragbarkeit auf Edge-Computing-Szenarien. Die Entwicklung erfolgte auf einem Apple-Silicon-(M3)-MacBook-Air mit 16 GB RAM unter Nutzung der PyTorch-MPS<sup>2</sup>-Beschleunigung, um eine effiziente Modellentwicklung zu ermöglichen. Als repräsentative Target-Hardware für das Embedded Deployment wurde der Raspberry

---

<sup>2</sup> Metal Performance Shaders. Diese dienen zur hardwarebeschleunigten Berechnung mit Geräten, die auf Apple-Silicon basieren.

Pi 5 (8 GB RAM) mit ARM Cortex-A76 CPU spezifiziert. Die Bilderfassung erfolgte über eine PiCamera 3 Wide mit Sony-IMX708-Sensor, die über die auf Raspberry-Pi-Systemen vorinstallierte PiCamera2-Bibliothek angesteuert wurde. Die Weitwinkel-Charakteristik ermöglichte die Erfassung eines erweiterten Sichtfeldes zur vollständigen Abdeckung von Fahrspurwechsel-Szenarien und seitlichen Verkehrsteilnehmern. Dadurch konnten kritische Fahrsituationen umfassender aufgenommen werden. Bei einer Bildauflösung von 1920 x 1080 Pixeln wurde eine fahrzeugspezifische Region of Interest definiert, um den Fokus auf verkehrsrelevante Bildsegmente zu legen.



Abbildung 1: Verkehrssituation mit markiertem relevantem Bildbereich

Die Fahrzeugdatenerfassung wurde über ELM327-kompatible Bluetooth-Adapter mit ISO-14230-Protokoll-Unterstützung an einem Audi A1 8X als Testfahrzeug realisiert. Als Basis für die OBD-II-Kommunikation wurde die Bibliothek „python-obd“ genutzt.

Für die Computer-Vision-Pipeline wurde YOLO12n als Baseline-Architektur gewählt. Als kleinstes verfügbares YOLO12-Modell mit knapp 2,55 Millionen Parametern stellt es einen Kompromiss zwischen Erkennungsgenauigkeit, Developer-Tooling und Inferenzgeschwindigkeit für Edge-Computing-Beschränkungen dar.

## 4. Datengrundlage und -preprocessing

Für die Entwicklung eines Machine-Learning-Modells zur Bremsereignis-Vorhersage, das mit Supervised Learning trainiert werden soll, sind präzise Ground-Truth-Labels erforderlich, die das tatsächliche Bremsverhalten des Fahrers dokumentieren. Da Standard-OBD-II-Parameter keine direkten Bremssignal-Informationen bereitstellen, wurde ein systematisches Reverse Engineering proprietärer Diagnosebefehle durchgeführt, um Zugang zu den erforderlichen Fahrzeugdaten zu erlangen.

### 4.1 OBD-II Reverse Engineering

#### 4.1.1 Grundlagen der OBD-II Kommunikation

Die On-Board-Diagnostics-II-Schnittstelle (OBD-II) stellt seit 1996 einen standardisierten Zugang zu Fahrzeugdiagnosedaten dar. Die Kommunikation erfolgt über verschiedene Protokolle, primär ISO 14230 [56] und ISO 15765 [57] (CAN). Moderne Fahrzeuge verwenden überwiegend das Controller Area Network (CAN)-Protokoll. Der Datenzugriff erfolgt über Parameter Identification Numbers (PIDs), die in verschiedenen Service-Modi organisiert sind.

Der Standard-OBD-II-Befehlssatz umfasst neun Service-Modi (Mode 01 bis Mode 09): Mode 01 ist für Live-Daten konzipiert und Mode 02 für eingefrorene Fehlerdaten. Diese Modi decken jedoch ausschließlich emissionsrelevante Parameter ab. Dadurch sind fahrdynamisch relevante Signale, wie beispielsweise Bremszustände, Lenkwinkel oder Beschleunigungssensoren, nicht direkt zugänglich.

#### 4.1.2 Proprietäre Diagnosemodi und Mode 22

Jenseits der standardisierten Service-Modi implementieren Fahrzeughersteller proprietäre Diagnosefunktionen, die erweiterte Parameterzugriffe ermöglichen. In Foren der „Car-Hacking“-Community findet man Beiträge, in denen beschrieben wird, wie mittels Mode 22 Daten gelesen und geschrieben werden konnten, die nicht standardmäßig im OBD-II-Protokoll zu finden sind. Diese Forenbeiträge beziehen sich meist auf Fahrzeuge der Hersteller Mazda und Ford. Allerdings konnte auch ein Beitrag gefunden werden, in dem mit Mode 22 Daten eines VW e-UP, also eines Fahrzeugs aus der Konzernfamilie des Audi A1 8X, gelesen werden konnten [58]. Mode 22 („Read Data

by Identifier") ist ein herstellerspezifischer Service, der Zugang zu internen Steuergerät-Parametern gewährt, die nicht in der Standard-OBD-II-Spezifikation enthalten sind. [58], [59], [60], [61]

Die systematische Exploration der Mode-22-Parameter-IDs erfolgte mittels eines automatisierten Scanning-Ansatzes. Der entwickelte Algorithmus iteriert über den gesamten 16-Bit<sup>3</sup>-PID-Adressraum (0x0000 bis 0xFFFF) und identifiziert gültige Parameter durch Auswertung der Steuergerät-Antworten.

```
def scan_mode_22_pids(obd_connection):
    valid_pids = []
    for pid in range(0x0000, 0xFFFF):
        hex_pid = f"{pid:04X}"
        cmd = obd.ObdCommand("CUSTOM_PID", f"Mode 22 PID {hex_pid}",
                             ("22" + hex_pid).encode(), 0, raw_string_decoder)
        response = obd_connection.query(cmd, force=True)
        if not response.is_null():
            valid_pids.append(hex_pid)
    return valid_pids
```

Abbildung 2: Abfrage-Code für OBD-II Mode 22 Addressraum

#### 4.1.3 Differentielle Analyse zur Bremssignal-Identifikation

Die Identifikation bremsspezifischer Parameter erfolgte durch eine vergleichende Analyse der Steuergerät-Datenströme vor und während stehender Bremsbetätigung in zwei distinkten Phasen:

1. **Baseline-Erfassung:** Dokumentation aller verfügbaren Mode-22-Parameter im unbetätigten Zustand.
2. **Stimulation-Response:** Erfassung derselben Parameter während aktiver Bremspedalbetätigung.
3. **differentielle Auswertung:** Identifikation signifikant veränderter Parameterwerte.

Diese Methodik identifizierte PID 0x3F9F als primären Kandidaten für bremsspezifische Signalisierung, da ausschließlich dieser Parameter konsistente Zustandsänderungen

---

<sup>3</sup> Mode 22 PIDs bestehen aus 2 Bytes (= 16 Bits), anders als beispielsweise Mode 01 PIDs

während Bremsvorgängen aufwies. Die systematische Exploration des gesamten Mode 22 Adressraums ergab keine weiteren fahrdynamisch relevanten Parameter, wodurch PID 0x3F9F als einzige verfügbare Quelle für Bremssignal-Informationen identifiziert wurde.

#### 4.1.4 Binäre Signaldekodierung und Protokollanalyse

Die Analyse der Rohdatenstruktur von PID 0x3F9F offenbarte eine byteorientierte Signalkodierung. Die Steuergerät-Antwort folgt dem Standard-OBD-Format mit einem 3-Byte Header (Service-Mode und PID) gefolgt von den eigentlichen Nutzdaten:

```
def decode_brake_signal(messages):
    d: bytes = messages[0].data
    d = d[3:] # Remove 1 mode and 2 PID bytes from response
    v = d[-1] & 0x01 # Extract least significant bit of last byte
    return bool(v)
```

Abbildung 3: Code für Dekodierung des Bremssignals

Die binäre Analyse ergab, dass das Bremssignal im niederwertigsten Bit (LSB) des letzten Nutzdaten-Bytes kodiert ist.

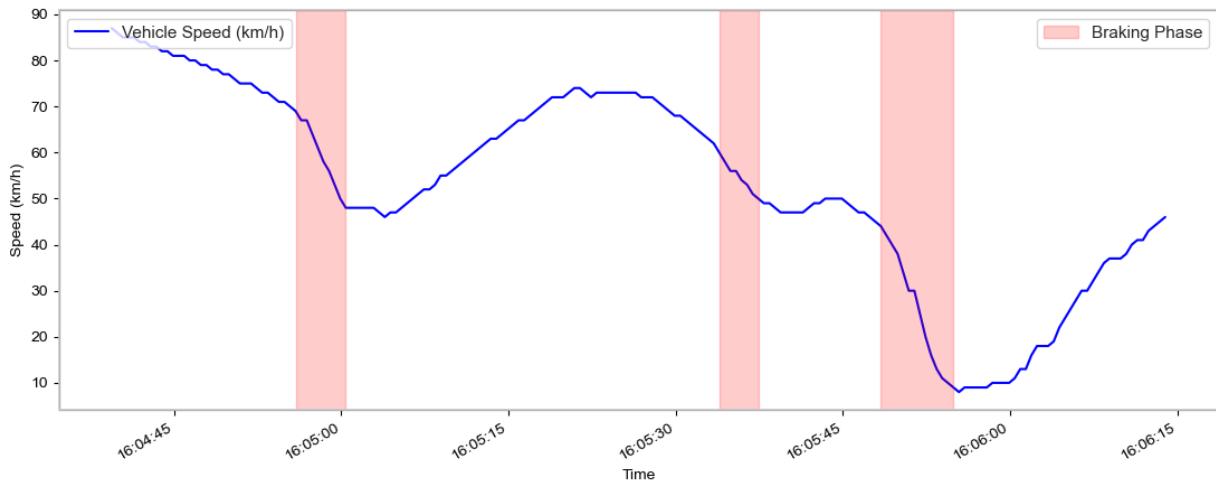
Bremszustand	Vollständige Response	Nutzdaten Byte	LSB (Bit 0)	Dekodiert
Nicht aktiv	0x223F9F00	0x00	0	False
Aktiv	0x223F9F01	0x01	1	True

Tabelle 1: Antwortdekodierung des Bremssignals

#### 4.1.5 Validierung und Korrelationsanalyse

Die statistische Auswertung von drei repräsentativen Testfahrten mit insgesamt 7.815 Datenpunkten über 65 Minuten ergab eine mittlere Korrelation von -0,38 ( $\sigma = 0,16$ ) zwischen Bremssignal-Aktivierung und Geschwindigkeitsverlauf. Der Korrelationsbereich von [-0,55, -0,22] zeigt eine konsistent negative Beziehung über alle Testfahrten hinweg, was die technische Validität des Reverse-Engineering-Ansatzes quantitativ bestätigt.

### Vehicle Telemetry Data Analysis - 2024-11-08\_16-00-48



*Abbildung 4: Fahrzeugelektrometrie mit Bremsphasen*

Die qualitative Analyse in Abbildung 4 des Zeitreihenverlaufs bestätigt die quantitativen Korrelationsergebnisse, indem eine deutliche visuelle Übereinstimmung zwischen aktivierten Bremsphasen und Geschwindigkeitsreduktionen erkennbar wird.

Das resultierende OBD-Command wird als BRAKE\_SIGNAL = OBDCommand("BRAKE\_SIGNAL", "Whether the braking signal is on", b"223F9F", 0, decode\_brake\_signal) als reguläres OBDCCommand-Objekt der python-obd Bibliothek in die Telemetrie-Pipeline integriert und ermöglicht die Echtzeit-Erfassung von Bremsereignissen [62].

#### **4.1.6 Sicherheitsaspekte und Risikominimierung**

Der Zugriff auf proprietäre Steuergerät-Parameter mittels Mode-22-Befehlen birgt potenzielle Risiken für die Fahrzeugsoftware, da bestimmte OBD-Befehle nicht nur lesend, sondern auch schreibend auf Steuergeräteparameter zugreifen können. Eine unsachgemäße Manipulation von Steuergerät-Konfigurationen kann zu Funktionsstörungen sicherheitsrelevanter Fahrzeugsysteme führen.

Zur Minimierung der Risiken wurde vor Beginn der Reverse-Engineering-Aktivitäten der Originalzustand aller relevanten Steuergeräte durch eine Fachwerkstatt dokumentiert und gesichert. Diese Vorsichtsmaßnahme gewährleistet die vollständige Wiederherstellbarkeit der Ausgangskonfiguration und reduziert das Risiko nachhaltiger Schäden an der Fahrzeugelektronik.

## 4.2 YOLO-Finetuning für das Autobahnszenario

Für die Implementierung eines leistungsfähigen Fahrzeugdetektionssystems zur Autobahnverkehrserkennung mussten vortrainierte YOLO-Modelle [63] systematisch evaluiert und optimiert werden. Generische Objektdetektoren, die auf allgemeinen Datensätzen wie COCO [64] trainiert wurden, zeigen in domänenspezifischen Anwendungsszenarien charakteristische Leistungsdefizite. Diese resultieren aus der statistischen Diskrepanz zwischen der Trainingsverteilung generischer Datensätze und den spezifischen visuellen Eigenschaften der Zieldomäne.

### 4.2.1 Empirische Baseline-Evaluation und Problemidentifikation

Die anfängliche Leistungsbewertung des vortrainierten YOLOv12n-Modells auf Basis von COCO offenbarte signifikante Defizite bei der Detektion in realen Fahrbedingungen. Insbesondere bei durch Fahrzeugvibrationen verursachter Bildunschärfe zeigte das Basismodell unzureichende Robustheit. Dies korreliert mit der Tatsache, dass die COCO-Trainingsdaten primär statische Szenen mit optimaler Bildqualität umfassen [64], während Anwendungen im Automotive-Bereich dynamische Umgebungen mit variierenden Bildqualitäten erfordern.

Eine systematische Analyse der Detektionsfehler identifizierte drei kritische Schwachstellen: Erstens erfolgt eine unzureichende Erkennung bei reduzierter Bildschärfe, zweitens ist die Performance bei großen Entfernungen, wie sie typischerweise auf Autobahnen vorkommen, suboptimal und drittens erfolgt eine inkonsistente Klassifikation zwischen verschiedenen Fahrzeugtypen unter variierenden Beleuchtungsbedingungen.

### 4.2.2 Datensatz-Analyse und Auswahlmethodik

Für das domänenspezifische Finetuning wurden zwei automobilbezogene Objekterkennungsdatensätze systematisch evaluiert: Nulimages [65] und Boxy [66]. Die Evaluationskriterien umfassten Domänen-Kompatibilität, Datenqualität, Hardware-Kompatibilität und Annotationsgenauigkeit.

**Nulimages-Datensatz:** Der Nulimages-Datensatz bietet 1,4 Millionen annotierte Bilder. Für die Implementierung wurden ausschließlich Aufnahmen der Front- und Rückkameras

(CAM\_FRONT, CAM\_BACK) extrahiert, da diese der Perspektive der installierten Fahrzeugkamera entsprechen. Aufnahmen der Seitenkameras wurden ausgeschlossen, um die Domänenkonsistenz zu gewährleisten. Die verfügbaren Fahrzeugklassen (vehicle.car, vehicle.truck, vehicle.motorcycle, vehicle.bus, vehicle.trailer) ermöglichen eine granulare Klassifikation bei gleichzeitig hoher Komplexität. [65]

**Boxy-Datensatz:** Der Boxy-Datensatz umfasst Autobahnverkehrsszenarien aus Kalifornien mit 34 charakteristischen Sequenzen. Ein wesentlicher technischer Vorteil liegt in den verfügbaren 3D-Bounding-Box-Annotationen, die neben konventionellen 2D-Begrenzungsrahmen auch spezifische „rear“-Annotationen für Fahrzeogrückseiten enthalten. Diese ermöglichen eine präzisere Schätzung der Fahrzeuggeometrie durch die explizite Erkennung der sichtbaren Fahrzeogrückseite. Dies ist für Abstandsberechnungen und die räumliche Orientierung in Autobahnszenarien von entscheidender Bedeutung. Basierend auf den im Boxy-Paper dokumentierten Metadaten wurden Nacht-Sequenzen systematisch ausgeschlossen, um konsistente Beleuchtungsbedingungen zu gewährleisten. Sequenzen mit schlechten Witterungsbedingungen und hoher Verkehrsichte wurden hingegen bewusst inkludiert, um die Modellrobustheit unter realistischen Bedingungen zu fördern. [66]

#### 4.2.3 Implementierung und Optimierungsresultate

Die Entscheidung für den Boxy-Datensatz basierte auf der optimalen Korrespondenz zwischen Aufnahmebedingungen und Zielszenario sowie der reduzierten Klassenkomplexität für die binäre Fahrzeugdetektion. Die YOLOv12n-Konfiguration wurde mit einer Trainingsauflösung von  $704 \times 704$  Pixeln implementiert, um eine Konsistenz zwischen Training und Inferenz zu gewährleisten und eine optimale Leistung auf der Zielhardware zu erreichen.

Als kritischer Optimierungsparameter erwies sich multi\_scale=True, welcher während des Trainings zufällige Skalierungen der Eingabebilder zwischen 50 % und 150 % der Basisauflösung anwendet [67]. Dieser Mechanismus verbessert die Detektionsleistung für weit entfernte Objekte signifikant, da das Modell lernt, Fahrzeuge bei verschiedenen Skalierungen zu erkennen [68] - ein essenzieller Aspekt für Autobahnszenarien mit großen Distanzvariationen. [69], [70]

Die empirische Evaluation demonstrierte eine substanzielle Verbesserung der Detektionsleistung gegenüber dem Basismodell, das auf dem COCO-Datensatz trainiert wurde [71]. Insbesondere die Robustheit gegenüber bewegungsinduzierter Bildunschärfe und die Erkennungsgenauigkeit bei großen Entfernungen zeigten messbare Verbesserungen.

## 4.3 Datenerfassung und -verarbeitung

Die Datenerfassung erfolgt durch das entwickelte DriveRecorder-System, das eine simultane Aufzeichnung von Kameraaufnahmen und OBD-II-Telemetriedaten ermöglicht. Das System verwendet eine Zwei-Thread-Architektur, um beide Datenströme parallel zu erfassen. Dadurch wird eine zeitliche Synchronisation mit einer Abweichung von weniger als 5 Millisekunden zwischen den Modalitäten erreicht.

Die Kamerakomponente verfügt über eine automatische Hardware-Erkennung, die zwischen dem Raspberry Pi Camera Module und Standard-USB-Webcams für Testzwecke unterscheidet. Für Raspberry-Pi-Systeme, die den Live-Betrieb widerspiegeln, werden spezifische Konfigurationsparameter wie Bildschärfe, Sättigung und angepasster Autofokus aktiviert. Die Standardauflösung beträgt 1920 x 1080 Pixel bei einer Bildwiederholrate von mindestens 30 FPS.

Die Telemetriedaten werden über eine OBD-II-Verbindung mit einer Abtastrate von ~8 Hz erfasst. Das System überwacht folgende Parameter: Fahrzeuggeschwindigkeit (SPEED), Motordrehzahl (RPM), Gaspedalposition (ACCELERATOR\_POS\_D), Motorlast (ENGINE\_LOAD) und das durch Reverse Engineering extrahierte binäre Bremssignal (BRAKE\_SIGNAL). Diese Werte werden laufend in einer Update-Loop aktualisiert und alle 0,5 Sekunden gemeinsam mit dem aufgenommenen Bild abgespeichert.

Zur Optimierung des Speicherbedarfs während der Trainingsdatenerfassung wird eine verlustbehaftete JPEG-Kompression mit einer Qualitätsstufe von 85 % angewendet. Ein negativer Einfluss auf die Objekterkennung durch YOLO wurde dabei nicht festgestellt. Diese Kompression reduziert den Speicherbedarf durchschnittlich um 69,3 %. In einer produktiven Real-Time-Pipeline würde sie jedoch nicht verwendet werden, da sie zusätzliche Latenz einführt, wie die folgende Grafik zeigt.

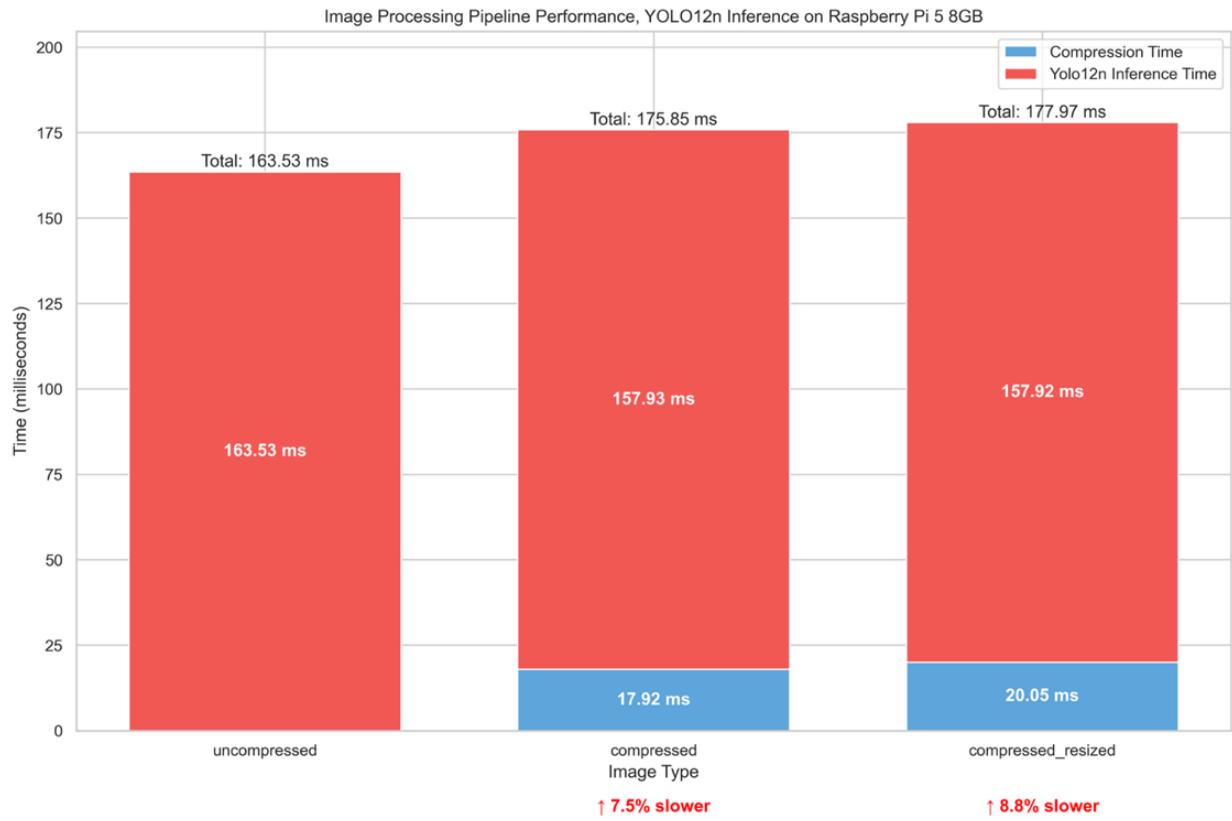


Abbildung 5: Performance bei Bildkompression

Zur Gewährleistung der zeitkritischen Datenerfassung mit 2Hz wurde eine Method-Caching-Strategie implementiert, bei der häufig verwendete Funktionen (`time.time`, `datetime.now`, `max`, `sleep`) in lokale Variablen zwischengespeichert werden. Diese Optimierung eliminiert wiederholte Namespace-Lookups während der Aufzeichnungsschleife und reduziert die Latenz zwischen Datenerfassung und Zeitstempel-Generierung [72]. Eigene Messungen zeigten, dass durch diese Implementierung die Abweichung vom Sollintervall von 500ms auf  $\pm 0,12\text{ms}$  begrenzt werden konnte, was einer Timing-Genauigkeit von 99,976% entspricht. Diese Präzision ist für die spätere Synchronisation zwischen Telemetrie- und Bilddaten sowie für die Erzeugung zeitlich konsistenter Trainingssequenzen essenziell.

## 4.4 Multimodales Feature Engineering

### Repräsentation externer Verkehrsteilnehmer

Die Objekterkennung externer Verkehrsteilnehmer erfolgt durch das gefinetunte YOLO-Modell und wird als strukturierte Merkmalssequenz für die multimodale Fusion aufbereitet. Jede Detektion wird durch einen sechsdimensionalen Merkmalsvektor  $d_i$  repräsentiert:

$$A_i = (x_2 - x_1) \cdot (y_2 - y_1)$$

$$d_i = [c_i, x_1, y_1, x_2, y_2, A_i]$$

wobei  $c_i$  die Konfidenz der Detektion,  $x_1, y_1, x_2, y_2$  die normalisierten Bounding-Box-Koordinaten und  $A_i$  die berechnete Objektfläche darstellen.

Zur Gewährleistung einheitlicher Eingabedimensionen werden pro Zeitschritt priorisiert nach Objektfläche maximal 12 Detektionen verarbeitet, wobei fehlende Detektionen durch Nullvektoren aufgefüllt und mittels boolescher Masken gekennzeichnet werden.

### Telemetriedaten

Die Fahrzeugtelemetrie wird über die implementierte OBD-II-Schnittstelle erfasst und umfasst sowohl standardisierte als auch proprietäre Diagnoseparameter:

**Geschwindigkeit  $v$ :** Fahrzeuggeschwindigkeit in km/h

**Motordrehzahl  $\omega$ :** Umdrehungen pro Minute

**Gaspedalposition  $\alpha$ :** Relative Position in Prozent

**Motorlast  $\lambda$ :** Relative Motorauslastung in Prozent

**Bremssignal  $\beta$ :** Binärer Zustand aus proprietärem Mode-22-Command

Die Rohdaten werden zur Gewährleistung numerischer Stabilität auf den Wertebereich [0,1] normalisiert. Für die kategoriale Gangwahlvariable wird eine One-Hot-Kodierung implementiert, die sechs Zustände (Leerlauf, Gang 1-5) abbildet.

### Gangwahlberechnung

Die Gangwahlberechnung basiert auf dem Verhältnis zwischen Fahrzeuggeschwindigkeit und Motordrehzahl, ergänzt durch Gaspedalposition und Motorlast zur Behandlung von

Sonderfällen. Die Implementierung verwendet vordefinierte Übersetzungsverhältnisse für das Fünfgang-Schaltgetriebe des Audi A1 8X.

Die Klassifikation erfolgt durch Vergleich mit den charakteristischen Übersetzungsverhältnissen:

$$r_{current} = \frac{v}{\omega}, \quad r_{gear} \in \{0.0095, 0.0165, 0.0275, 0.0405, 0.0535\}$$

Zur Kompensation von Messungenauigkeiten wird eine adaptive Fehlertoleranz implementiert:

$$\epsilon(r) = \epsilon_{base} + r \cdot \epsilon_{prog}$$

mit  $\epsilon_{base} = 0.001$  und  $\epsilon_{prog} = 0.025$ .

Diese progressive Toleranz berücksichtigt, dass die Fahrzeuggeschwindigkeit als Ganzzahl-Wert übertragen wird, was bei höheren Geschwindigkeiten zu größeren relativen Schwankungen im berechneten Übersetzungsverhältnis führt.

### Bremskraftberechnung

Die Bremskraftschätzung wurde als präventives Feature-Engineering-Element implementiert, um zukünftige Erweiterungen des Copiloten-Systems um weitere fahrdynamische Parameter zu ermöglichen. Die Berechnung findet während der Fahrt statt, da zu diesem Zeitpunkt höhere Abtastraten zu den Steuergeräten möglich sind.

Die Schätzung der relativen Bremskraft erfolgt durch eine zeitbasierte Analyse der Geschwindigkeitsverläufe in einem zweistufigen Verfahren:

#### Schritt 1: Erfassung der Referenzverzögerung

Zur Etablierung einer fahrzeugspezifischen Baseline wird die Geschwindigkeitsänderung in einem Zeitfenster von 3 Sekunden vor Aktivierung des Bremssignals erfasst. Dieses Zeitfenster berücksichtigt natürliche Geschwindigkeitsschwankungen durch Verkehrsfluss, Steigungen, Motorbremse oder aerodynamischen Widerstand.

#### Schritt 2: Messung der Bremsverzögerung

Während des aktiven Bremsvorgangs wird die Geschwindigkeitsänderung in einem Zeitfenster von 1,15 Sekunden kontinuierlich analysiert. Die kürzere Zeitspanne ermöglicht eine responsive Erfassung der tatsächlichen aktuellen Bremsintensität.

Die gewählten Zeitfenster-Längen wurden empirisch durch systematische Analyse der aufgezeichneten Telemetriedaten-Verläufe bestimmt und repräsentieren einen Kompromiss zwischen statistischer Robustheit und zeitlicher Auflösung.

Die Bremskraftschätzung basiert auf der numerischen Approximation der Geschwindigkeitsableitung durch lineare Regression:

**Basisverzögerung (vor dem Bremsen):**

$$a_{baseline} = \frac{dv}{dt} \mid t \in [t_{brake} - 3s, t_{brake}]$$

**Aktuelle Verzögerung (während des Bremsens):**

$$a_{current} = \frac{dv}{dt} \mid t \in [t_{brake}, t_{brake} + 1.15s]$$

**Relative Bremskraft:**

$$F_{brake\_rel} = -1 \times (a_{current} - a_{baseline})$$

Die resultierenden Werte werden im nachgelagerten Trainingsprozess durch den datensatzspezifischen Maximalwert normiert.

**Implementierungsdetails** Die Geschwindigkeitsgradienten werden mittels gewichteter linearer Regression berechnet, wobei neuere Datenpunkte höher gewichtet werden. Dies reduziert den Einfluss von Messrauschen und verbessert die Robustheit gegenüber kurzfristigen Geschwindigkeitsschwankungen. Die lineare Regression ist insbesondere notwendig, da die Geschwindigkeitswerte über OBD-II als Integer-Werte übertragen werden und somit diskrete Stufen aufweisen. Durch die Regression über mehrere Datenpunkte wird der kontinuierliche Geschwindigkeitstrend approximiert und eine präzisere Gradientenschätzung ermöglicht.

Die Verwendung der Differenzbildung zwischen aktueller und Basisverzögerung eliminiert systematische Einflüsse wie Fahrzeugmasse, Straßenneigung oder Rollwiderstand, die in beiden Zeitfenstern gleichermaßen wirken.

## **4.5 Datensatzcharakteristiken**

### **Aufnahmebedingungen und Szenario-Abdeckung**

Der Datensatz umfasst 12 vollständig verarbeitete Aufzeichnungssitzungen mit einer Gesamtlänge von 43.104 individuellen Frames, erfasst über einen Zeitraum von sechs Monaten (Dezember 2024 bis Mai 2025). Sämtliche Aufnahmen wurden unter realen Autobahnbedingungen auf den Streckenabschnitten A8 Stuttgart-Augsburg, A6 Stuttgart-Ansbach sowie A9 Ingolstadt-München durchgeführt, wobei die synchronisierte Erfassung von Kameradaten (Full HD, 1920×1080 Pixel) und OBD-II-Telemetriedaten mit einer einheitlichen Sampling-Rate von 2 Hz erfolgte.

Die Aufnahmebedingungen variieren systematisch bezüglich Verkehrsdichte, Tageszeit und Witterungsbedingungen. Zur Gewährleistung konsistenter Analysebedingungen wurden Szenarien mit Baustellen, extremen Stausituationen und Tunneldurchfahrten systematisch ausgeschlossen. Die zeitliche Verteilung der verbleibenden Aufzeichnungen zeigt eine Abdeckung verschiedener Verkehrsszenarien: von dichtem Berufsverkehr bis zu geringem Verkehrsaufkommen während Randzeiten. Die geografische Konsistenz durch ausschließliche Autobahnaufnahmen gewährleistet eine homogene Umgebungscharakteristik bezüglich Straßengeometrie und Verkehrsregeln.

### **Sequenzgenerierung und Kontinuitätsfilterung**

Die temporale Segmentierung der Rohdaten wurde unter Anwendung eines Sliding-Window-Verfahrens mit einer Sequenzlänge von 20 Frames ( $\triangleq$  10 Sekunden bei 2 Hz Sampling-Rate) durchgeführt. Zur Sicherstellung der zeitlichen Integrität wurden ausschließlich kontinuierliche Sequenzen verwendet, wobei Diskontinuitäten mit Zeitlücken  $> 525$  ms systematisch ausgeschlossen wurden. Durch den Einsatz dieses Filters wurde die Anzahl der verwertbaren Sequenzen von den theoretisch möglichen Sliding-Window-Kombinationen auf 42.686 Sequenzen reduziert.

### **Kritische Situationsverteilung**

Die Analyse der Verteilung der Ereignisse offenbart die charakteristische Imbalance-Struktur realer sicherheitskritischer Datensätze. Von den 42.686 generierten Sequenzen sind 1.200 Sequenzen (2,8 %) als positive Bremsereignisse klassifiziert, was einem

Verhältnis von 1:36 entspricht. Coasting-Ereignisse<sup>4</sup> (Gaspedalposition < 7,5 %) treten mit einer Frequenz von ca. 3.050 Ereignissen (7,1 %) signifikant häufiger auf.

Die Klassenimbalance für Bremsereignisse liegt im erwarteten Bereich für Autobahnszenarien, da diese Situationen naturgemäß selten auftreten im fließenden Autobahnverkehr. Diese Verteilung erfordert spezielle Behandlung durch gewichtete Loss-Funktionen und klassengewichtete Metriken, die Precision-Recall-Charakteristika gegenüber der Accuracy priorisieren.

### **Ground Truth Validierung**

Die Gewinnung des Ground Truth für Bremsereignisse erfolgte durch das in Kapitel 4.1 beschriebene OBD-II Reverse Engineering Verfahren.

### **Datenaufteilung und Split-Strategie**

Die Aufteilung der Datensätze erfolgte aufzeichnungsbasiert, um eine strikte Vermeidung von Data Leakage zwischen Training und Evaluation zu gewährleisten. Die Zuordnung der Aufzeichnungssitzungen zu den Splits wurde durch eine stratifizierte Verteilung vorgenommen, die sowohl die Anzahl der Bremsereignisse als auch die Gesamtsequenzanzahl berücksichtigt:

Split	Zielauflistung	Aufzeichnungen	Sequenzen	Bremssequenzen
Train	75 %	9 (75 %)	30.043 (70,4 %)	866 (72,1 %)
Validation	15 %	1 (8,3 %)	8.328 (19,5 %)	225 (18,8 %)
Test	10 %	2 (16,7 %)	4.315 (10,1 %)	109 (9,1 %)
Datensatz	100 %	12	42.686	1.200

Tabelle 2: Datensatzaufteilung

Die aufzeichnungsbasierte Aufteilung gewährleistet, dass zeitlich benachbarte Sequenzen derselben Fahrt nicht gleichzeitig in Training und Evaluation verwendet werden. Das Bremsereignis-Verhältnis zeigt eine konsistente Verteilung über alle Splits ( $2,7 \% \pm 0,2 \%$ ), was die Repräsentativität der Evaluationsergebnisse sicherstellt.

---

<sup>4</sup> Ereignisse, in denen vom Gas gegangen wird

## 5. Modellarchitektur und Implementierung

### 5.1 Gesamtarchitektur und Datenfluss

Das entwickelte multimodale System implementiert eine dreistufige Pipeline-Architektur zur zeitkritischen Bremsereignis-Vorhersage. Die Architektur folgt dem in Abschnitt 3.1 etablierten modularen Design-Prinzip und kombiniert die Vorteile interpretierbarer Komponententrennung mit End-to-End-Optimierbarkeit.

#### 5.1.1 Dreistufige Verarbeitungsarchitektur

Die Systemarchitektur gliedert sich in drei sequenzielle Transformationsstufen, die gemeinsam trainiert und optimiert werden können:

**Input Encoder Module (E):** Die erste Verarbeitungsstufe transformiert heterogene Eingabemodalitäten in einheitliche Embedding-Repräsentationen. Telemetriedaten und Objektdetektionen werden separat verarbeitet und in einen gemeinsamen d-dimensionalen Embedding-Raum projiziert:

$$H_{tel}, H_{det} = E(X_{telem}, X_{vision}, M)$$

mit  $H_{tel} \in R^{B \times T \times d}$  und  $H_{det} \in R^{B \times T \times N \times d}$ .

**Fusion Module (F):** Die zweite Stufe kombiniert die modalitätsspezifischen Embeddings durch verschiedene Fusionsstrategien. Das Modul aggregiert zunächst die variable Anzahl von Objektdetektionen pro Zeitschritt und fusioniert anschließend die Modalitäten:

$$H_{fused} = F(H_{tel}, H_{det}, M)$$

wobei  $H_{fused} \in R^{B \times T \times d_{out}}$  die Repräsentation mit Ausgabedimension  $d_{out}$  darstellt.

**Output Decoder (D):** Die finale Verarbeitungsstufe transformiert die fusionierten Features in aufgabenspezifische Vorhersagen für multiple Zeithorizonte.<sup>5</sup> Das Modul verarbeitet die temporale Sequenzinformation und generiert binäre Klassifikationen:

$$Y = D(H_{fused})$$

---

<sup>5</sup> Beispielsweise brake\_1s ( $\hat{=}$  Vorhersage, ob in einer Sekunde gebremst werden muss)

### 5.1.2 Temporale Sequenzverarbeitung

Das System operiert auf zeitlich strukturierten Datensequenzen mit einer standardisierten Sequenzlänge von  $T = 20$  Zeitschritten, entsprechend 10 Sekunden Fahrzeughistorie bei 2Hz Abtastrate. Die temporale Struktur ermöglicht die Erfassung fahrdynamischer Trends und Verkehrssituationsentwicklungen, die für die Bremsereignis-Vorhersage erforderlich sind.

Jeder Zeitschritt  $t$  enthält synchronisierte multimodale Observationen:

- Fahrzeugtelemetrie:  $x_{tel}^{(t)} \in R^{D_{tel}}$
- Objektdetektionen:  $x_{det\_i}^{(t)}$  für  $i = 1, \dots, N_t$  mit  $N_t \leq N$
- Gültigkeitsmaske:  $m^{(t)} \in \{0,1\}^N$

Die variable Anzahl gültiger Detektionen  $N_t \leq N$  pro Zeitschritt wird durch Zero-Padding auf die maximale Detektionsanzahl  $N = 12$  normalisiert und mittels boolescher Masken gekennzeichnet.

### 5.1.3 End-to-End-Optimierung trotz Modularität

Trotz der expliziten Komponententrennung ermöglicht die Architektur eine gemeinsame Optimierung aller Parameter  $\theta = \theta_{\mathcal{E}}, \theta_{\mathcal{F}}, \theta_{\mathcal{D}}$  durch eine einheitliche Verlustfunktion:

$$L(\theta) = w_h \cdot BCE(y_h, \hat{y}_h)$$

mit horizontspezifischen Gewichtungen  $w_h$  für die gewichtete binäre Cross-Entropy-Verlustfunktion über alle Vorhersagehorizonte  $h \in \{1s, 2s, 3s, 4s, 5s\}$ .

Diese Implementierung folgt dem von Tesla etablierten Paradigma modularer End-to-End-Systeme, welches die Vorteile interpretierbarer Komponentenarchitekturen mit holistischer Gradientenoptimierung kombiniert [1], [73]. Für die Konzeption des gesamten Modells wurde auf vordefinierte Machine-Learning-Module von PyTorch [74] zurückgegriffen.

## 5.2 Input Encoder Implementierung

Die Input Encoder-Schicht transformiert die heterogenen Eingabemodalitäten (Telemetrie und Objektdetektionen) in einen einheitlichen Embedding-Raum. Die modulare Architektur ermöglicht den systematischen Vergleich verschiedener Verarbeitungsstrategien für multimodale Fahrzeugdaten.

**Architekturnprinzip:** Telemetriedaten und Objektdetektionen werden durch separate neuronale Transformationen verarbeitet, ohne direkten Informationsaustausch zwischen den Modalitäten auf Encoder-Ebene.

**Mathematische Formulierung:** Für eine Eingabesequenz der Länge  $T$  wird die Telemetrie  $X_{tel} \in R^{T \times d_{tel}}$  durch eine lineare Projektion transformiert:

$$H_{tel} = \text{LayerNorm}(X_{tel}W_{tel} + b_{tel})$$

wobei  $W_{tel} \in R^{d_{det} \times d_{emb}}$  die Gewichtsmatrix und  $d_{emb}$  die Embedding-Dimension darstellt.

Um granularere Objektrepräsentationen zu erhalten, werden Objektdetektionen  $X_{det} \in R^{T \times N_{max} \times d_{det}}$  mittels einer zweistufigen linearen Transformation verarbeitet:

$$H_{det} = \text{LayerNorm}(\text{ReLU}(X_{det}W_{det1} + b_{det1})W_{det2} + b_{det2})$$

mit  $W_{det1} \in R^{d_{det} \times 2d_{emb}}$  und  $W_{det2} \in R^{2d_{emb} \times d_{emb}}$ .

**Normalisierung und Regularisierung:** Layer Normalization wird nach jeder Transformation angewendet, um Input-Verteilungen der Aktivierungsfunktionen zu stabilisieren. Dropout mit Wahrscheinlichkeit  $p_{drop} = 0.15$  reduziert Overfitting-Risiken.

**Output-Charakteristika:** Der Encoder gibt individuelle Objekt-Embeddings ohne interne Aggregation zurück. Dies ermöglicht nachgelagerten Fusionsmodulen maximale Flexibilität bei der Informationskombination. Die Ausgabe erfolgt als Dictionary mit separaten Tensoren für Telemetrie-Features ( $H_{tel}$ ), Detection-Features ( $H_{det}$ ) und der Validitätsmaske für die Objektdetektionen. Die vorliegende Vorgehensweise zur Datenkodierung ist als eher simpel zu erachten. Im Gesamtmodell wird demnach eine Early-Fusion-Strategie<sup>6</sup> verfolgt.

---

<sup>6</sup> Vgl. Kapitel 2.2.4

## 5.3 Fusion Module Implementierung

Die Fusionsschicht kombiniert die separat kodierten Modalitäten zu einer einheitlichen Repräsentation für die nachfolgende Sequenzverarbeitung. Die modulare Implementierung ermöglicht den systematischen Vergleich verschiedener Fusionsstrategien zwischen den Modalitäten.

**Architekturprinzip:** Objektrepräsentationen werden durch statistische Aggregation auf Zeitpunktebene reduziert, anschließend mit der Telemetrierepräsentation konkateniert und durch nichtlineare Transformation verarbeitet. Das Modul implementiert eine deterministische Aggregation ohne parametergesteuerte Attention-Mechanismen.

**Mathematische Formulierung:** Für eine Eingabesequenz<sup>7</sup> mit Telemetrie-Features  $f_{tel} \in R^{T \times d_{emb}}$ , Objektdetections  $f_{det} \in R^{T \times N_{max} \times d_{emb}}$  und Validitätsmaske  $M \in 0,1^{T \times N_{max}}$  erfolgt die Fusion durch:

$$f_{agg}^{(t)} = \frac{\sum_{n=1}^{N_{max}} M^{(t,n)} \cdot f_{det}^{(t,n)}}{\sum_{n=1}^{N_{max}} M^{(t,n)} + \epsilon}$$

$$f_{concat}^{(t)} = [f_{tel}^{(t)} ; f_{agg}^{(t)}] \in R^{2 \cdot d_{emb}}$$

$$f_{fused}^{(t)} = LayerNorm\left(ReLU\left(f_{concat}^{(t)} \cdot W + b\right)\right)$$

wobei  $W \in R^{2 \cdot d_{emb} \times d_{out}}$  die Gewichtsmatrix der linearen Transformation darstellt und  $\epsilon = 10^{-8}$  numerische Stabilität gewährleistet.

**Normalisierung und Regularisierung:** Layer Normalization wird nach der MLP-Transformation angewendet, um Input-Verteilungen der Aktivierungsfunktionen zu stabilisieren. Dropout mit Wahrscheinlichkeit  $p_{drop} = 0.15$  reduziert Overfitting-Risiken.

**Output-Charakteristika:** Das Fusionsmodul generiert eine uniforme Sequenzrepräsentation  $f_{fused} \in R^{T \times d_{out}}$  mit  $d_{out} = 2 \cdot d_{emb}$  als Standardkonfiguration.

---

<sup>7</sup> aus einem Encoder-Modul

## 5.4 Output Decoder Architekturen

Die Output Decoder-Schicht transformiert fusionierte multimodale Features in aufgabenspezifische Prädiktionen für verschiedene Zeithorizonte. Zwei fundamentale Verarbeitungsparadigmen werden implementiert: sequenzielle LSTM-basierte und parallele Transformer-basierte Architektur. Die detaillierten Architektspezifikationen der verwendeten PyTorch-Module nn.LSTM [75] und nn.TransformerEncoder [76] sind in der offiziellen PyTorch-Dokumentation dokumentiert. Im Folgenden werden die charakteristischen Eigenschaften und Implementierungsaspekte der jeweiligen Ansätze analysiert.

### 5.4.1 Gemeinsame Multi-Task Klassifikationsstruktur

Beide Decoder-Varianten implementieren identische, auf die jeweilige Aufgabe abgestimmte Klassifikationsköpfe für die parallele Prädiktion von Bremsereignissen. Für jeden konfigurierten Zeithorizont  $\tau$  wird ein separater binärer Klassifikator definiert:

$$MLP_\tau: R^{(d_{hidden})} \rightarrow R^1$$

$$MLP_\tau = \text{Linear}(\text{Dropout}(\text{ReLU}(\text{Linear}(h_{final}))))$$

Die Ausgabe erfolgt als Logits vor Sigmoid-Aktivierung zur numerischen Stabilität während des Trainings. Konfigurationsparameter umfassen  $d_{hidden} = 128$  und Dropout-Wahrscheinlichkeit  $p_{drop} = 0.25$  zwischen den linearen Schichten.

### 5.4.2 LSTMOutputDecoder

**Verarbeitungscharakteristik:** Das LSTM-Modul verarbeitet Eingabesequenzen zeitschrittweise durch rekurrente Hidden States. Der finale Hidden State  $h_T$  komprimiert die gesamte Sequenzinformation als und dient als Basis für nachgelagerte Klassifikation.

**Architektspezifikation:** Zweischichtige LSTM-Architektur mit unidirektionaler Verarbeitung zur Erhaltung kausaler Zeitabhängigkeiten. Die Konfiguration implementiert num\_layers = 2, hidden\_size = 128 und bidirectional = False. Dropout wird zwischen LSTM-Schichten mit  $p_{drop} = 0.25$  angewendet.

#### Charakteristische Eigenschaften:

Sequenzielle Verarbeitung: Zeitschritt t+1 abhängig von t

Speichereffizienz:  $O(T)$  Speicherkomplexität bezüglich Sequenzlänge

Verarbeitungslatenz: Lineare Abhängigkeit von Sequenzlänge  $T$

### 5.4.3 TransformerOutputDecoder

**Verarbeitungscharakteristik:** Das Transformer-Modul nutzt Self-Attention-Mechanismen für simultane Verarbeitung aller Zeitpunkte ohne rekurrente Abhängigkeiten. Die finale Token-Repräsentation  $F^{enc_T}$  dient als Sequenzaggregation für nachgelagerte Klassifikation.

**Architekturnspezifikation:** Vierschichtige Transformer-Encoder-Architektur mit Multi-Head Self-Attention. Die Konfiguration implementiert  $\text{num\_layers} = 4$ ,  $d_{\text{model}} = 128$ ,  $\text{num\_heads} = 8$  und  $d_{\text{feedforward}} = 256$ .

#### Charakteristische Eigenschaften:

Parallele Verarbeitung: Simultane Betrachtung aller Zeitpunkte

Speicherkomplexität:  $O(T^2)$  durch Attention-Matrizen

Verarbeitungslatenz: Konstante Parallelisierbarkeit unabhängig von  $T^8$

### 5.4.4 Architektur-Vergleich

**Verarbeitungsparadigma:** LSTM implementiert inhärent kausale Zeitreihenverarbeitung mit sequenzieller Abhängigkeitsstruktur. Transformer ermöglichen eine bidirektionale Kontexterfassung durch globale Attention-Mechanismen.

**Effizienz:** Die LSTM-Architektur zeigt lineare Speicherskalierung und begrenzte Parallelisierbarkeit. Transformer-Architektur erfordert quadratische Speicherkomplexität, ermöglicht jedoch vollständige Parallelisierung auf moderner GPU-Hardware.

**Anwendungskontext:** Die LSTM-Verarbeitung bietet Vorteile für Echtzeitanwendungen mit begrenzten Ressourcen durch reduzierte Speicheranforderungen. Die Transformer-Architektur eignet sich hingegen für Anwendungen mit verfügbarer paralleler Rechenkapazität und Anforderungen an die globale Erfassung von Sequenzkontexten.

---

<sup>8</sup> Unter der Voraussetzung, dass die Hardware-Architektur eine solche Verarbeitung ermöglicht

# **6. Experimentelle Evaluation**

## **6.1 Experimenteller Aufbau und Methodik**

Die experimentelle Evaluation erfolgt auf dem in Kapitel 4.5 beschriebenen Datensatz aus zwölf Autobahnfahrten. Die extreme Klassenimbalance mit Bremsereignissen im Verhältnis 1:36 und Rollereignissen im Verhältnis 1:14 spiegelt realistische Fahrbedingungen wider, in denen Ereignisse, die Verzögerung benötigen, naturgemäß selten auftreten.

Als Baseline-Architektur dient eine Kombination aus SimpleInputEncoder und SimpleConcatenationFusion, welche die grundlegende Funktionalität der modularen Architektur ohne komplexe Attention-Mechanismen demonstriert. Die systematische Evaluation fokussiert sich auf den Vergleich zwischen LSTM- und Transformer-basierten Output-Decodern bei gleichbleibendem Encoder- und Fusionsmodul, um den isolierten Einfluss der Sequenzmodellierung zu quantifizieren.

Die Bewertung erfolgt primär über Precision-Recall Area Under Curve (PR-AUC) als Hauptmetrik, da diese bei extremer Klassenimbalance aussagekräftiger ist als Accuracy-basierte Metriken. Ergänzend werden ROC-AUC sowie F1-Scores bei optimal bestimmten Schwellwerten berichtet. Die Schwellwertoptimierung erfolgt separat für verschiedene Zielfunktionen (F1-Score, Matthews Correlation Coefficient, Balanced Accuracy) zur Berücksichtigung unterschiedlicher Anwendungsszenarien. [77], [78], [79]

## **6.2 Loss-Funktion Evolution und Task-Balancing**

### **6.2.1 Weighted Binary Cross-Entropy**

Zur systematischen Analyse der Vorhersagefähigkeit wurden zunächst separate Single-Task-Modelle für Brems- und Rollereignisse mit beiden Decoder-Architekturen (LSTM, Transformer) trainiert. Diese Baseline-Experimente verwendeten gewichtete binäre Kreuzentropie zur initialen Behandlung der Klassenimbalance.

Die empirischen Ergebnisse demonstrieren einen konsistenten Leistungsabfall bei längeren Vorhersagehorizonten über alle Tasks hinweg. Für Bremsereignisse zeigt sich

eine dramatische Performance-Verschlechterung von PR-AUC = 0,342 (1s) auf 0,041 (5s), entsprechend einem Leistungsabfall von 88%, wie die folgende Grafik zeigt.

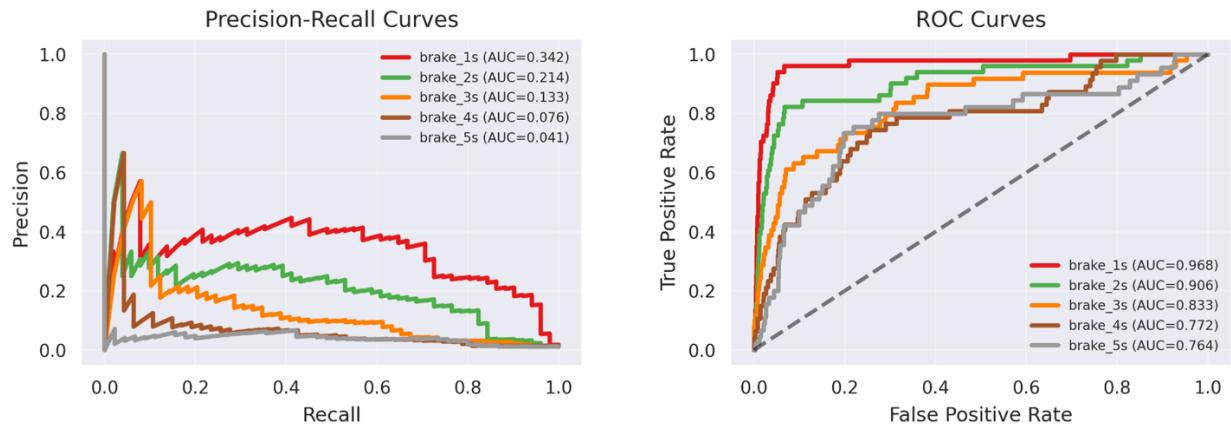


Abbildung 6: PR- und ROC-Kurven bei Bremsvorhersage

Rollereignisse weisen eine ähnliche, jedoch weniger ausgeprägte Degradation auf: von PR-AUC = 0,732 (1s) auf 0,204 (5s), entsprechend 72% Leistungsabfall, wie die folgende Grafik zeigt.

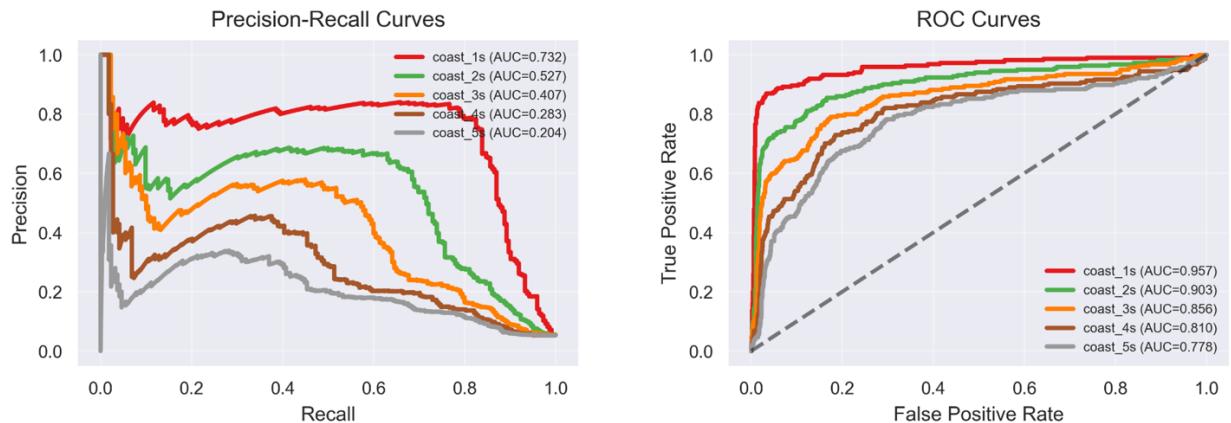


Abbildung 7: PR- und ROC-Kurven bei Rollvorhersage

Basierend auf dieser systematischen Analyse erfolgte die empirische Task-Auswahl zur Fokussierung auf die Vorhersagehorizonte 1s und 2s. Diese Entscheidung resultiert aus dem exponentiellen Performance-Abfall bei längeren Horizonten und entspricht praktischen Anforderungen präventiver Fahrerassistenzsysteme, die primär kurzfristige Prädiktion erfordern.

## 6.2.2 Focal Loss Implementation

Zur präzisen Behandlung der task-spezifischen Klassenimbalance wurde Focal Loss [80] implementiert. Die mathematische Formulierung lautet:

$$FL(p_t) = -\alpha_t(1 - p_t) \gamma \log(p_t)$$

wobei  $p_t$  die Wahrscheinlichkeit der korrekten Klasse,  $\alpha_t$  der klassenbalancierende Gewichtungsfaktor und  $\gamma$  der Fokussierungsparameter zur Unterdrückung einfacher Beispiele darstellt. Einfache Beispiele sind dabei Klassifikationsfälle mit hoher Vorhersagewahrscheinlichkeit, die das Training dominieren würden.

Die task-spezifische Konfiguration wurde über systematische Hyperparameter-Evaluierungen optimiert und stellt die beste empirisch ermittelte Parametrisierung dar:

Task	Verhältnis Pos:Neg	$\alpha$	$\gamma$	Task-Gewicht	Begründung
brake_1s	1:36	0,2	3,0	1,0	Extreme Imbalance, safety-critical
brake_2s	1:36	0,2	2.75	0,95	Extreme Imbalance, geringere Priorität
coast_1s	1:14	0,3	2.0	0,85	Moderate Imbalance, Effizienz-orientiert
coast_2s	1:14	0,3	1.75	0,8	Moderate Imbalance, längerer Horizont

Tabelle 3: Final beste Focal-Loss-Konfiguration

Eine Analyse diverser Trainingszyklen offenbart, dass die Implementierung von Focal Loss tendenziell zu einer erhöhten Anzahl korrekter Identifikationen von Brems- und Rollereignissen (erhöhte True Positive Rate) führt, jedoch zu einer marginalen Steigerung der False Positive Rate. Obwohl die Anzahl der falsch-positiven Klassifikationen geringfügig zunimmt, werden insgesamt deutlich bessere Gesamtergebnisse erzielt. Dies ist auf die verbesserte Sensitivität für seltene, sicherheitskritische Ereignisse zurückzuführen, welche den Anstieg unkorrekter Vorhersagen überkompensiert.

## 6.3 Architektur-Vergleich: LSTM vs. Transformer Decoder

### 6.3.1 Performance-Analyse

Die final beste Modellkonfiguration implementiert einen SimpleInputEncoder (Embedding-Dimension: 64), eine SimpleConcatenationFusion (Ausgabedimension: 128) und einen 4-Layer Transformer Decoder mit Multi-Head Attention (8 Heads, Hidden-Dimension: 128, Dropout: 0,15).

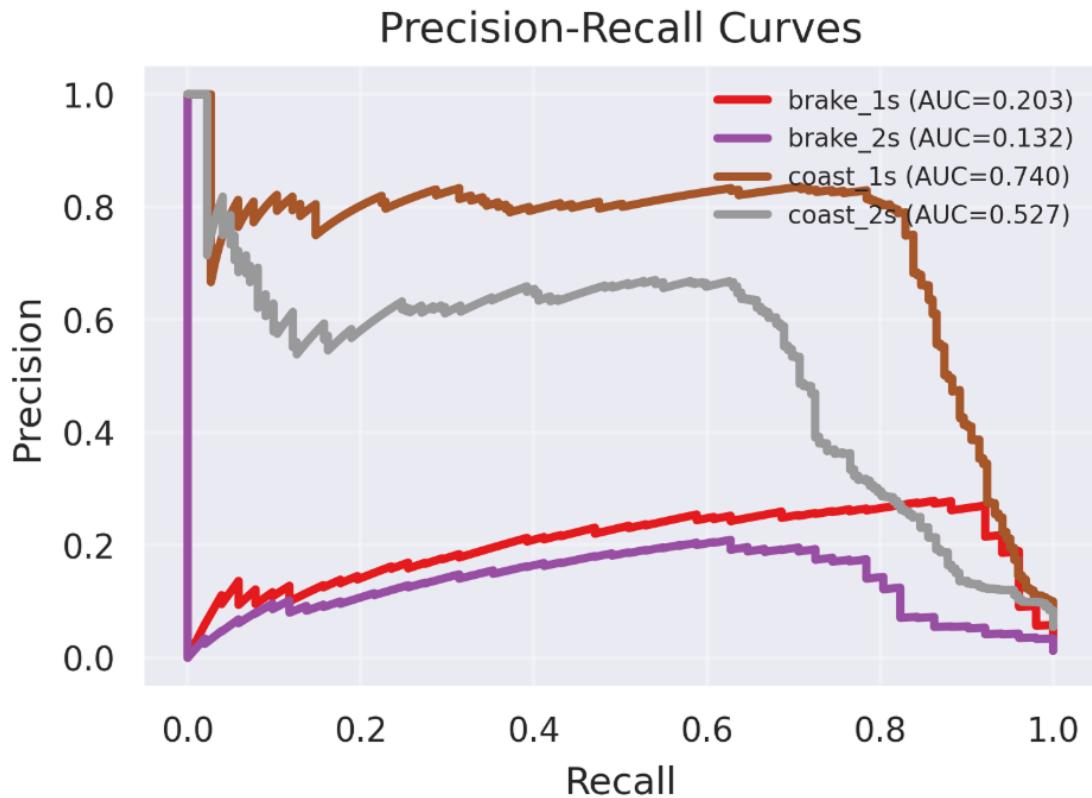


Abbildung 8: Precision-Recall-Kurve bei Brems- und Rollvorhersage

Die Precision-Recall Kurven zeigen deutliche Performance-Unterschiede zwischen den Tasks. Der sicherheitskritische Task `brake_1s` erreicht eine PR-AUC von 0,203, was einer 7,25-fachen Verbesserung gegenüber dem Random Baseline (0,028) entspricht. Die Coast-Tasks demonstrieren höhere absolute Performance-Werte mit PR-AUC-Werten von 0,740 (`coast_1s`) und 0,527 (`coast_2s`), reflektieren jedoch die geringere Klassenimbalance dieser Tasks.

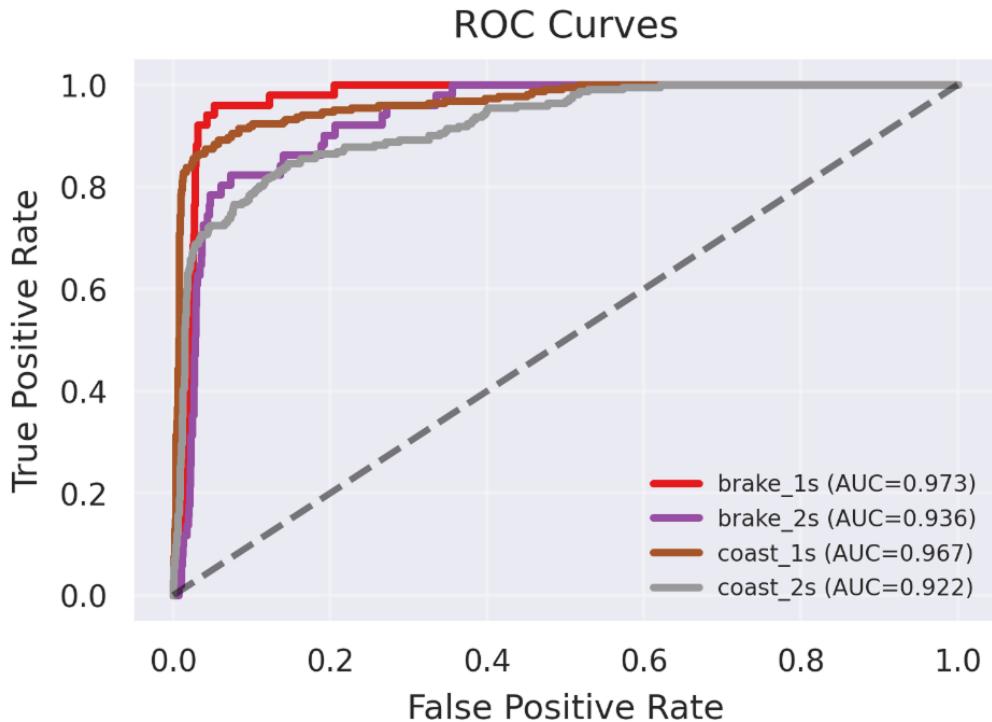


Abbildung 9: ROC-Kurve bei Brems- und Rollvorhersage

Die ROC-Analyse bestätigt die Diskriminierungsfähigkeit des Modells. Alle Tasks zeigen ROC-AUC-Werte oberhalb von 0,9, wobei coast\_1s (0,967) und coast\_2s (0,922) die stärkste Separierung zwischen den Klassen aufweisen. Die Bremsereignis-Tasks erreichen ROC-AUC-Werte von 0,973 (brake\_1s) und 0,936 (brake\_2s).

### 6.3.2 Qualitative Decoder-Charakteristika

Der LSTM-Decoder zeigte erhöhte Instabilität während des Trainings, manifestiert durch inkonsistente Konvergenz über verschiedene Loss-Konfigurationen. Die Architektur demonstrierte verstärkte Overfitting-Tendenz bei der extremen Klassenimbalance, resultierend in deutlich erhöhten False-Positive-Raten für Bremsereignisse.

Die Verwechslungsmatrizen dokumentieren die praktische Performance-Charakteristik des Transformer-Decoders, der als bestes Modell ausgewählt wurde. Für brake\_1s zeigt die Matrix eine True Negative Rate von 96,8% bei einer True Positive Rate von 92,2%. Das Modell klassifiziert 97,1% der negativen brake\_2s-Instanzen korrekt bei 58,8% korrekter Positivklassifikation. Die Coast-Tasks erreichen höhere Ausgewogenheit mit True Positive Rates von 81,2% (coast\_1s) und 63,1% (coast\_2s).

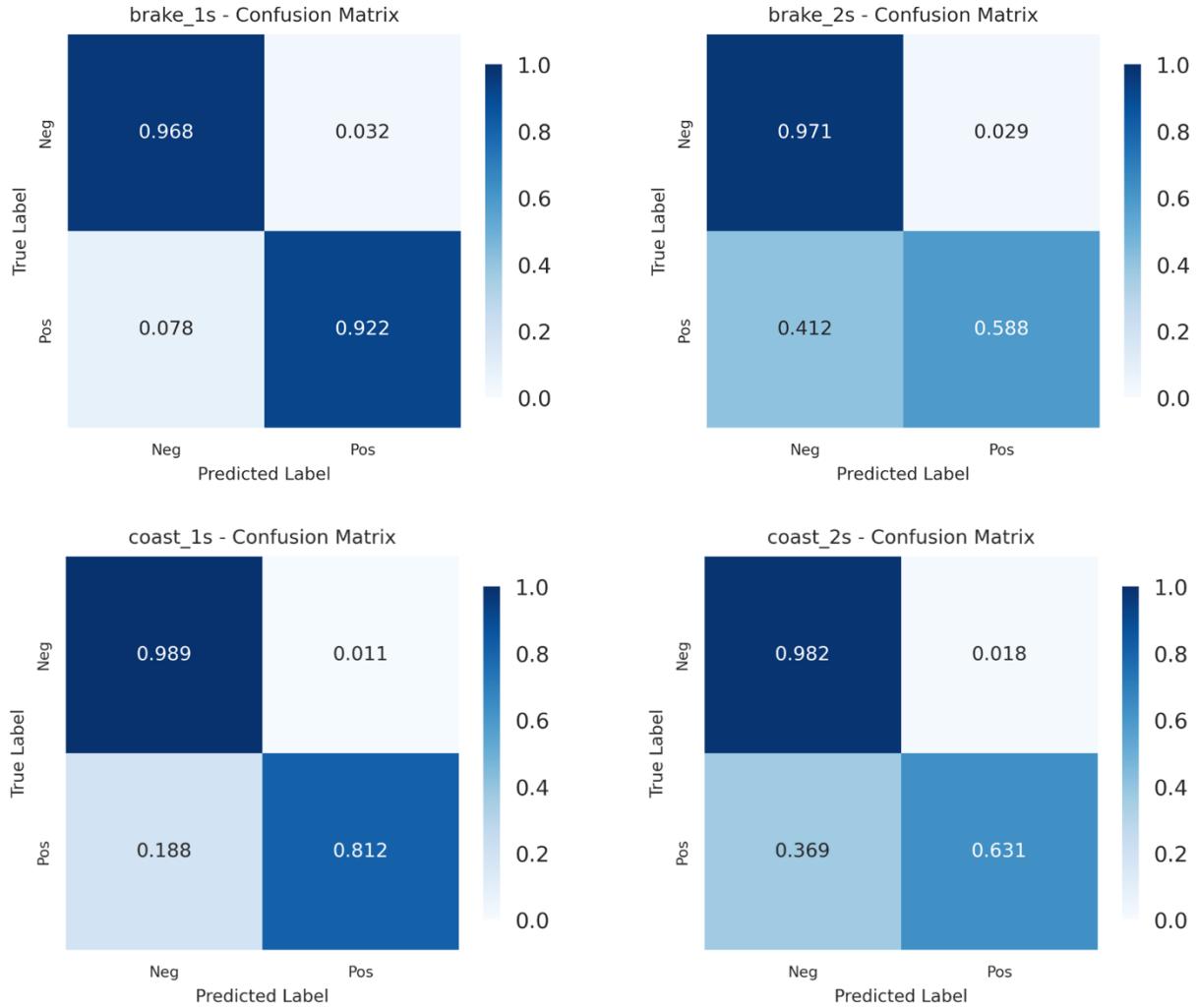


Abbildung 10: Normalisierte Verwechslungsmatrizen für Brems- und Rollvorhersagen

Der Transformer-Decoder demonstriert konsistenter Generalisierungseigenschaften über alle Tasks hinweg. Die Architektur zeigt reduzierte Hyperparameter-Sensitivität und stabilere Konvergenz-Charakteristik. Die finale Modellselektion basierte auf der optimierten Performance des sicherheitskritischen Tasks `brake_1s` unter Minimierung des Leistungsabfalls bei sekundären Vorhersage-Tasks. Experimentelle Evaluationen mit erhöhten Embedding-Dimensionen resultierten in gesteigerter Parameteranzahl ohne korrespondierende Performance-Verbesserung, wodurch die Konfiguration mit 64-dimensionalen Embeddings das optimale Verhältnis zwischen Modellkomplexität und Effizienz unter den gegebenen Latenz-Constraints repräsentiert.

## 6.4 Transformer Attention Muster

### 6.4.1 Gelernte zeitliche Abhangigkeiten

Die Analyse der temporalen Attention-Muster des TransformerOutputDecoders zeigt konsistente Spezialisierung auf zeitlich jungere Informationen.

Alle vier Transformer-Layer entwickeln einen ausgepragten Recency Bias mit Attention-Gewichten von 0,36-0,44 fur die letzten finf Sequenz-Positionen, verglichen mit 0,14-0,18 fur die ersten finf Positionen. Diese asymmetrische Gewichtung reflektiert die zeitkritische Natur der Bremsereignis-Pradiktion, bei der aktuelle Fahrsituationen hohere pradiktive Relevanz besitzen als historische Zustande.

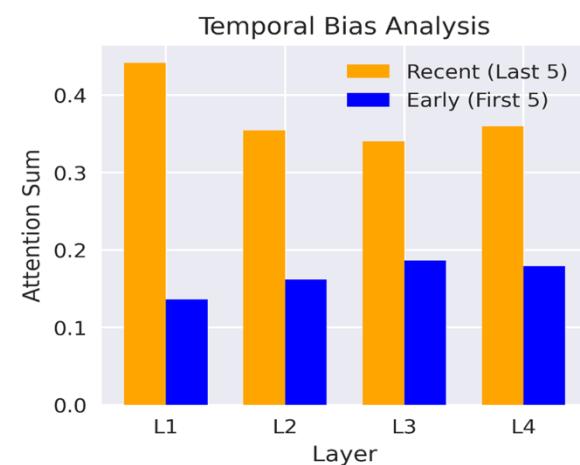


Abbildung 11: Recency Bias fur Sequenzmodellierung bei Transformer

### 6.4.2 Temporale Sequenzmodellierung

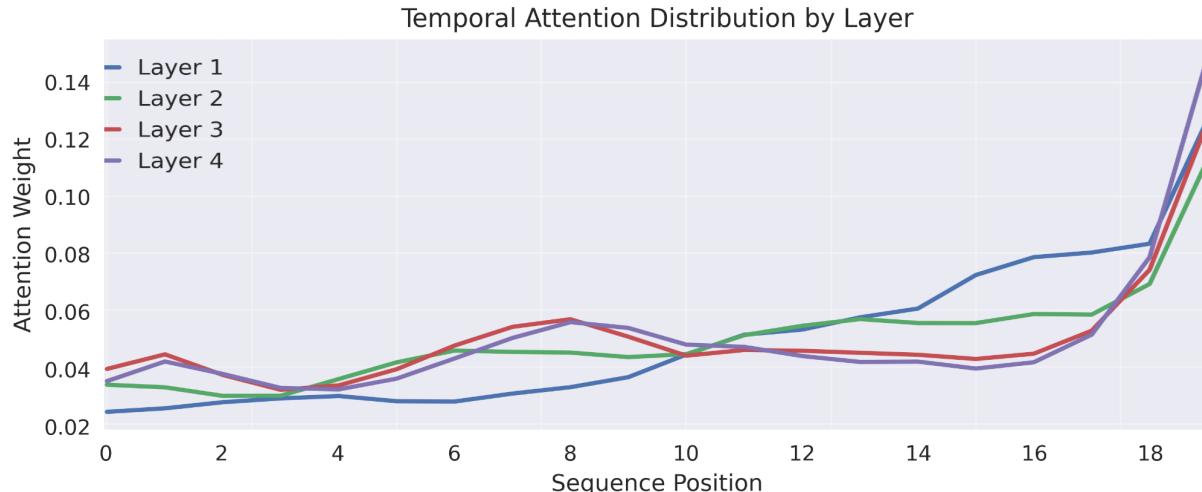


Abbildung 12: Attention-Muster fur Sequenzmodellierung bei Transformer

Samtliche Transformer-Layer konvergieren auf Position 19 als primare Attention-Peak, was einer direkten Fokussierung auf den finalen Zeitschritt entspricht. Diese Konzentration auf die unmittelbare Gegenwart erklart die vergleichbare Performance zum

LSTMOutputDecoder, da beide Architekturen effektiv eine zeitlich gewichtete Aggregation mit Recency Bias implementieren.

Empirische Tests mit fünf Transformer-Layern im Decoder-Modul zeigten keine Performance-Verbesserung und erhöhte Overfitting-Tendenz, was die Redundanz zusätzlicher Attention-Schichten bestätigt. Die quantifizierte Recency-Bias-Dominanz validiert die Hypothese, dass für Bremsereignis-Prädiktion primär lokale temporale Kontexte relevant sind, nicht weitreichende Sequenzdependenzen.

## 6.5 Edge Computing Performance

Die Evaluierung der Inferenz-Performance stellt einen kritischen Aspekt für die praktische Implementierung multimodaler Fahrerassistenzsysteme auf Embedded-Hardware dar. Zur systematischen Analyse der Hardware-Anforderungen wurden die entwickelten Modellarchitekturen unter verschiedenen Optimierungsstrategien auf dem Raspberry Pi 5 mit 8GB RAM untersucht.

### Experimenteller Aufbau

Die Performance-Evaluation erfolgte durch Messung der durchschnittlichen Inferenzzeiten über 1.000 Iterationen für jede Architektur-Optimierungs-Kombination. Als Testdatensatz dienten realistische Eingabesequenzen, die beim Live-Betrieb anfallen würden.

Drei Optimierungsstrategien wurden systematisch evaluiert:

- **Baseline:** Standard PyTorch-Implementierung ohne spezifische Optimierungen
- **PyTorch Compile:** Anwendung von `torch.compile()` für Just-In-Time Kompilierung
- **Mixed Precision:** Verwendung von FP16-Arithmetik zur Speicher- und Laufzeitoptimierung

## Architekturvergleich

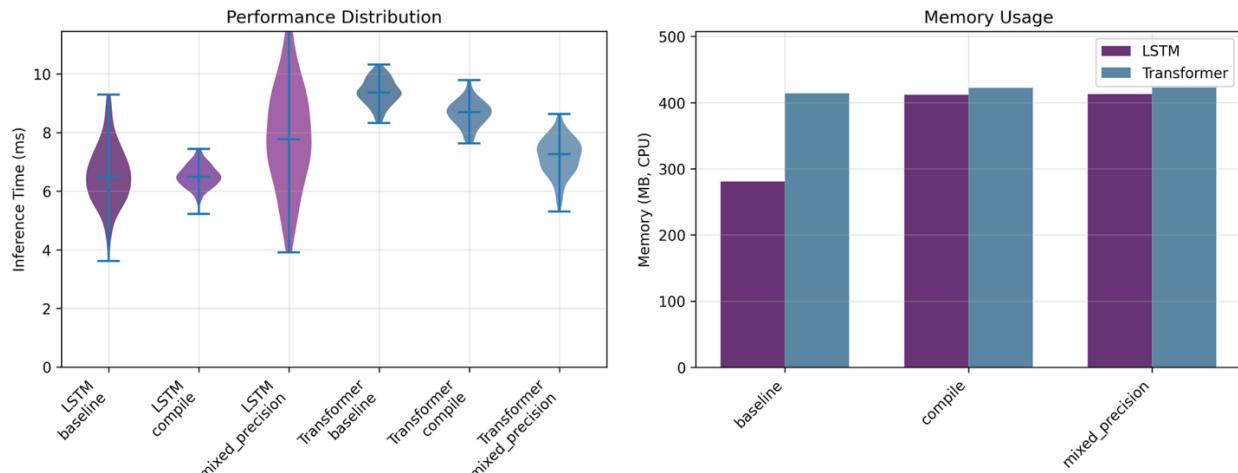


Abbildung 13: Leistungs- und Speicherverbrauchsmessung des multimodalen Modells

Die quantitative Analyse in Abbildung 13 zeigt systematische Performance-Unterschiede zwischen den Decoder-Architekturen bei identischen Eingabedimensionen:

Optimierung	LSTM Latenz (ms)	Transformer Latenz (ms)	LSTM Speicher (MB)	Transformer Speicher (MB)
Baseline	$6,51 \pm 1,07$	$9,28 \pm 0,43$	280	413
PyTorch Compile	$6,47 \pm 0,39$	$8,47 \pm 0,46$	412	422
Mixed Precision	$7,50 \pm 1,84$	$7,17 \pm 0,69$	412	422

Tabelle 4: Inferenzzeiten des multimodalen Modells

Die Modellkomplexität unterscheidet sich signifikant zwischen den Architekturen:

Architektur	Gesamtparame	Parameter Decoder-Modul	Komplexitätsverhältnis
LSTM	588.676	561.668	1,00
Transformer	853.380	826.372	1,45

Tabelle 5: Parameteranzahl verschiedener Modellarchitekturen

Die LSTM-Architektur weist niedrigere Inferenzzeiten bei geringerem Speicherverbrauch auf. In der Baseline-Konfiguration korrelieren sowohl Latenz- als auch Speicherunterschiede direkt mit der Parameteranzahl: Der Transformer erreicht eine um 43 % höhere Latenz (9,28ms vs. 6,51ms) und einen um 46 % höheren Speicherverbrauch

(413MB vs. 280MB), was dem 1,45-fachen Komplexitätsverhältnis entspricht. Bei angewendeten Optimierungen schwächt sich diese Korrelation jedoch ab.

## Optimierungseffektivität

Abbildung 14 quantifiziert die Effektivität der angewendeten Optimierungsstrategien durch Speedup-Faktoren relativ zur Baseline-Performance:

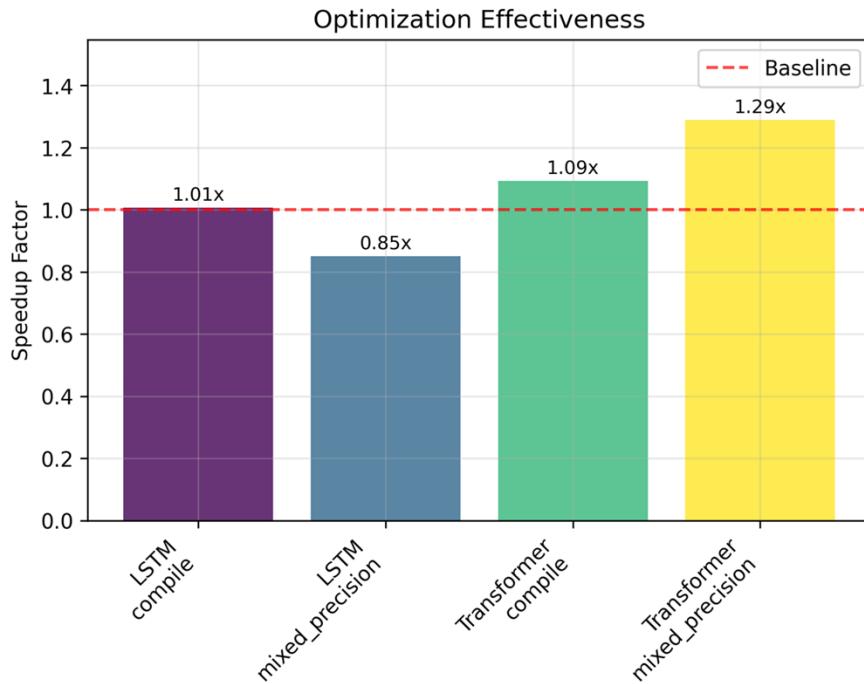


Abbildung 14: Performanceverbesserung der Modellarchitekturen nach Optimierungen

Die Compile-Optimierung erzielt minimale Performance-Verbesserungen mit Speedup-Faktoren von 1,01 (LSTM) bzw. 1,09 (Transformer). Diese resultiert aus der Transformation der PyTorch-Modelle in optimierte Rechengraphen mittels Dynamo, die durch Backend-Compiler wie TorchInductor für ARM-spezifische Architekturen optimiert werden. Die resultierende Kernel-Fusion und Minimierung des Python-Overheads führt zu den beobachteten Latenz-Reduktionen. [81], [82]

Die Mixed-Precision-Optimierung zeigt architekturabhängige Ergebnisse: Während LSTM eine Performance-Degradation (Slowdown-Faktor 0,85) erfährt, erzielt Transformer signifikante Verbesserungen (Speedup-Faktor 1,29). Diese Differenz resultiert aus der unterschiedlichen Parallelisierbarkeit: Die Self-Attention-Operationen

des Transformers profitieren von FP16-Matrixmultiplikationen, während die LSTM-Sequenzverarbeitung durch den FP16-Konvertierungs-Overhead beeinträchtigt wird.

### Edge Computing Implikationen

Die Gesamtpipeline-Latenz ergibt sich aus der sequenziellen Verkettung von Sensordatenabfrage von OBD-II und Kamera, YOLO-Fahrzeugerkennung und multimodalem Modell zur Verzögerungsvorhersage.

Pipeline-Komponente	Latenz (ms)	Anteil Gesamtlatenz
Datenabfrage	< 2,5	2,25 %
YOLO-Modell	122,58	92,35 %
Multimodales Modell	7,17 (Transformer, MP) <sup>9</sup>	5,40 %
Gesamtpipeline	132,75	100%

Tabelle 6: Latenzaufteilung der Gesamtpipeline

Die YOLO-Komponente dominiert mit einem Anteil von >92 % die Gesamtlatenz und stellt den primären Engpass für die Echtzeit-Leistung dar. Trotz des marginalen LSTM-Geschwindigkeitsvorteils ist in Produktionsumgebungen die Transformer-Architektur vorzuziehen, da sie eine überlegene Klassifikationsleistung, stabilere Inferenzzeiten und einen vernachlässigbaren Anteil an der Gesamtpipeline-Latenz (< 3ms Differenz) bietet.

Die Speicheranforderungen von maximal 430 MB (multimodal) + ~960 MB (YOLO) = ~1.390 MB bleiben deutlich unter der 8-GB-Kapazität des Raspberry Pi 5. Die Gesamtpipeline erfüllt damit die in Kapitel 3.1 definierten Latenzziele.

---

<sup>9</sup> Multimodales Modell zur Verzögerungsvorhersage, das die Transformer-basierte Sequenzmodellierung und Mixed Precision-Optimierung nutzt

## 6.6 Einfluss von Sequenzüberlappung

Die Wahl des Sequence Stride-Parameters beeinflusst sowohl die Datensatz-Größe als auch die Generalisierungsfähigkeit des Modells. Der Stride definiert den Versatz zwischen aufeinanderfolgenden Sliding-Window-Sequenzen und bestimmt damit den Überlappungsgrad zwischen benachbarten Trainingssamples. Zur Optimierung dieser Hyperparameter wurden systematische Trainings und Evaluierungen mit Stride-Werten von 1, 2, 3 und 4 Frames durchgeführt.

Die empirischen Ergebnisse zeigen, dass Stride 1 (maximale Überlappung) die beste Modellperformance erzielte, obwohl dies der theoretischen Erwartung widerspricht. Typischerweise wird ein höherer Stride-Wert (geringere Überlappung) bevorzugt, da dies die relative Diversität der Trainingssequenzen erhöht und Overfitting auf temporale Korrelationen reduziert [83]. Ein Stride von 4 Frames würde beispielsweise die Anzahl der Trainingssequenzen um den Faktor 4 reduzieren, gleichzeitig aber die Unabhängigkeit zwischen den Samples erhöhen.

Die Überlegenheit von Stride 1 in diesem Kontext ist primär auf die begrenzte Datenverfügbarkeit zurückzuführen. Bei einem Gesamtdatenvolumen von 42.626 Sequenzen aus 12 Aufzeichnungen war die maximale Datenausnutzung entscheidender als die theoretischen Vorteile reduzierter zeitlicher Korrelation zwischen den Sequenzen. Die signifikante Asymmetrie der Klassenverteilung (2,8 % Bremseignisse) potenziert diesen Effekt, da jede verfügbare positive Trainingsinstanz von entscheidender Relevanz für die Modellkonvergenz ist.

Diese Beobachtung verdeutlicht eine fundamentale Limitation des vorliegenden Datensatzes: Die Datenknappheit erfordert aggressive Datenausnutzungsstrategien, die unter Umständen zu schlechtere Generalisierungsfähigkeit führen können. Für produktive Anwendungen wäre eine Erweiterung des Datensatzes erforderlich, um robuste Stride-Konfigurationen mit reduzierten temporalen Abhängigkeiten zu ermöglichen.

# 7. Diskussion

## 7.1 Methodische Erkenntnisse

### 7.1.1 Modularer Architektur-Ansatz

Die systematische Evaluation von Encoder-Fusion-Decoder-Kombinationen bestätigt die Effektivität der modularen Architekturphilosophie als Methodik für die multimodale Systemanalyse. Durch die Isolation einzelner Systemkomponenten ist eine kontrollierte Quantifizierung spezifischer Architekturentscheidungen ohne verzerrende Einflussfaktoren möglich. Diese Methodik stellt somit einen systematischen Ansatz zur Bewertung komplexer Deep-Learning-Architekturen dar.

Der kontrollierte Vergleich zwischen LSTM- und Transformer-basierten Decodern bei identischen Input-Encoder- und Fusion-Komponenten demonstriert die praktische Anwendbarkeit modularer Evaluation. Die Transformer-basierte Sequenzmodellierung zeigt eine konsistente Überlegenheit hinsichtlich der Trainingsstabilität und des Konvergenzverhaltens bei extremer Klassenimbalance. Somit etabliert sich der Transformer als bevorzugte Architektur für diese sicherheitskritische Anwendung mit asymmetrischen Klassenverteilungen.

### 7.1.2 Sequenzmodellierung und Hardwarelimitierungen

Die Analyse temporaler Attention-Muster offenbart fundamentale Erkenntnisse über gelernte Repräsentationen in zeitkritischen Vorhersagesystemen. Die konsistente Entwicklung eines Recency Bias über alle Transformer-Layer hinweg zeigt, dass aktuelle Zustandsinformationen prädiktive Überlegenheit gegenüber historischen Kontexten besitzen. Diese Beobachtung validiert die Hypothese der zeitkritischen Relevanz für die Prädiktion von Bremsereignissen.

Die empirische Evaluation von Hardware-Optimierungsstrategien demonstriert die Dominanz architekturspezifischer Parallelisierbarkeit gegenüber theoretischer Parametereffizienz. Die Mixed-Precision-Optimierung kehrt die erwartete Performance-Hierarchie zwischen LSTM und Transformer um, was die Bedeutung einer an die Deployment-Umgebung angepassten Architektur-Evaluation unterstreicht. Diese

Erkenntnis etabliert Hardware-Charakteristika somit als kritischen Konstruktionsparameter für echtzeitfähige Deep-Learning-Systeme.

## 7.2 Technische Limitationen

Die entwickelte multimodale Architektur zur Echtzeiterkennung kritischer Fahrsituationen unterliegt mehreren systemimmanenten Beschränkungen, die sowohl die Generalisierbarkeit als auch die praktische Anwendbarkeit der Ergebnisse einschränken. Diese Limitationen lassen sich in drei wesentliche Kategorien unterteilen: Umgebungsmodellierung, Datensatzcharakteristika und Hardwarelimitierungen.

### 7.2.1 Umgebungsmodellierung

Die implementierte Wahrnehmungsarchitektur basiert auf einer Kamera mit  $60^\circ$  FOV, was eine fundamentale Limitation gegenüber modernen  $360^\circ$ -Surround-View-Systemen darstellt. Diese Beschränkung führt zu einer unvollständigen Umgebungsrepräsentation, da kritische Verkehrsteilnehmer außerhalb des Sichtfelds nicht erfasst werden können. Im Gegensatz zu produktiven ADAS-Systemen, die typischerweise Multiple-Sensor-Fusion einsetzen, reduziert sich die Situationswahrnehmung auf den frontalen Bereich.

Das YOLO-Modell wurde ausschließlich auf dem Boxy-Dataset für Fahrzeugerkennung optimiert, wodurch weitere Beschränkungen entstehen. Die Klassifikation erfolgt fahrzeugklassenunabhängig und beschränkt sich auf Fahrzeugheckansichten. Relevante Verkehrselemente wie Verkehrsschilder, Fahrbahnmarkierungen oder andere Verkehrsteilnehmer werden nicht modelliert. Diese Limitation reduziert die Kontextinformationen für die Vorhersage kritischer Situationen erheblich.

Die temporale Modellierung erfolgt frame-basiert ohne explizite Bewegungsschätzung oder Tracking-Komponenten. Moderne Ansätze wie Tesla's Hydranet implementieren räumlich-zeitliche Kohärenz durch Bird's-Eye-View-Repräsentationen und Multi-Frame-Fusion. Die vorliegende Architektur verzichtet auf derartige Mechanismen, wodurch Bewegungsmuster und Trajektorien nur implizit durch die Decoder-Module der entwickelten Architektur erfasst werden.

## 7.2.2 Datensatz

Der Trainingsdatensatz weist mehrere fahrzeug- und umgebungsspezifische Beschränkungen auf. Die OBD-II-Implementierung ist ausschließlich auf den Audi A1 8X kalibriert, insbesondere bezüglich der proprietären Bremssignal-Extraktion über Mode 22 Commands. Die abgeleiteten Merkmale wie Gangberechnung und Bremskraftschätzung basieren auf fahrzeugspezifischen Parametern (Übersetzungsverhältnisse, Motorcharakteristika), was die Übertragbarkeit auf andere Fahrzeugklassen einschränkt.

Mit lediglich 12 Aufnahmen und circa 6 Stunden verwertbarem Material ist der Datensatz quantitativ limitiert. Die extreme Klassenimbalance von 1:36 für Bremsereignisse reflektiert sowohl die Datensatzgröße als auch die Beschränkung auf Autobahnszenarien. Urbane Fahrsituationen, Baustellen, Tunnel oder Nachtfahrten sind nicht repräsentiert, wodurch die Generalisierbarkeit auf diverse Fahrbedingungen nicht gewährleistet ist.

## 7.2.3 Hardwarelimitierungen

Die Implementierung auf dem Raspberry Pi 5 ohne dedizierten AI-Beschleuniger stellt eine signifikante Einschränkung der Rechenleistung dar. Die ARM Cortex-A76 CPU unterstützt keine hardwarebeschleunigte INT8-Matrix-Multiplikation<sup>10</sup> [84], wodurch Quantisierungsoptimierungen nicht vollständig ausgeschöpft werden können. Die AI-Hat+-Erweiterung wurde aus Ressourcengründen nicht integriert, was eine zusätzliche Inferenzbeschleunigung verhindert.

Im Vergleich zu Teslas FSD-Hardware 4.0 mit 50 TOPS Rechenleistung [85] erreicht das Raspberry-Pi-System lediglich 1-2 TOPS CPU-Leistung [86]. Während das Perception Model von Tesla acht Kameras mit mindestens 36 Hz verarbeitet [1], [18], limitiert die YOLO-Inferenz allein das System auf etwa 8 Hz<sup>11</sup>. Diese Leistungsdiskrepanz verdeutlicht die Differenz zwischen Forschungsprototyp und produktionsreifer Hardware. Die Echtzeitfähigkeit wird durch diese Hardware-Beschränkungen fundamental eingeschränkt.

---

<sup>10</sup> Matrixmultiplikation auf Basis von 8 Bit-Ganzzahlen

<sup>11</sup> Vgl. Kapitel 6.5

## **8. Fazit und Ausblick**

### **8.1 Wissenschaftliche Beiträge**

Die vorliegende Arbeit entwickelt und evaluiert einen multimodalen Machine-Learning-Ansatz zur Echtzeiterkennung kritischer Fahrsituationen für präventive Fahrerassistenzsysteme. Die systematische Integration von Fahrzeugtelemetrie und visueller Wahrnehmung schafft eine methodische Grundlage für die Vorhersage von Bremsereignissen unter realistischen Hardware-Bedingungen.

#### **Technische Innovationen**

Die Arbeit leistet mehrere substanziale technische Beiträge zur automobilen KI-Forschung. Das OBD-II Reverse Engineering etabliert eine reproduzierbare Methodik zur Extraktion proprietärer Fahrzeugdaten durch systematische Mode 22 Command-Exploration. Die identifizierte Befehlssequenz 223F9F ermöglicht erstmals den direkten Zugriff auf Audi-spezifische Bremssignale über OBD-II.

Die entwickelte modulare Architekturphilosophie stellt eine methodische Innovation für die Evaluation multimodaler Deep-Learning-Systeme dar. Die systematische Trennung von Input-Encoder, Fusion-Module und Output-Decodern ermöglicht eine kontrollierte Quantifizierung architekturnspezifischer Leistungsunterschiede ohne konfundierende Variablen. Diese Methodik überwindet die in der Literatur häufig anzutreffende Vermischung unterschiedlicher Designentscheidungen in monolithischen Architekturvergleichen.

#### **Praktische Realisierung**

Der entwickelte Prototyp demonstriert die Machbarkeit multimodaler Fahrzeugsysteme unter stringenten embedded Constraints. Die vorliegende Arbeit demonstriert die prinzipielle Umsetzbarkeit echtzeitfähiger Wahrnehmungssysteme abseits hochperformanter GPU-Infrastrukturen durch die Integration von YOLO-basierter Objekterkennung mit telemetriegestützter Sequenzmodellierung auf einem Raspberry Pi 5. Die implementierte Pipeline zeichnet sich durch eine vollständige Integration aus, welche den gesamten Prozess von der Sensordatenakquisition bis zur Bremsereignisprädiktion abdeckt.

Die systematische Architektur-Evaluation liefert empirische Evidenz für die Überlegenheit der Transformer-basierten Sequenzmodellierung gegenüber den LSTM-Ansätzen in sicherheitskritischen Klassifikationsaufgaben mit extremer Klassenimbalance. Diese Erkenntnis trägt zur methodischen Fundierung von Architekturentscheidungen automobiler KI-Systeme bei, wobei die Faktoren Trainingsstabilität und Konvergenzverhalten von besonderer Relevanz sind.

### **Wissenschaftliche Einordnung**

Die Arbeit positioniert sich im Kontext moderner automotive AI-Systeme als Forschungsprototyp mit bewussten Simplifikationen. Im Vergleich zu produktiven ADAS-Systemen von Tesla, MobilEye, Bosch oder Waymo implementiert der entwickelte Ansatz eine reduzierte Sensorkonfiguration und fokussiert auf fundamentale multimodale Fusion-Mechanismen. Diese Reduktion ermöglicht eine isolierte Analyse der Kernkomponenten, ohne die Komplexität vollständiger Produktionssysteme zu berücksichtigen.

Die methodische Kontribution besteht in erster Linie in der systematischen Evaluation modularer Architekturen für die multimodale Fahrzeugwahrnehmung. Obwohl bereits bestehende Arbeiten häufig spezifische Architekturkombinationen evaluieren, etabliert diese Arbeit eine reproduzierbare Methodik zur kontrollierten Bewertung unterschiedlicher Designentscheidungen. Es lässt sich konstatieren, dass die vorliegende Methodik prinzipiell auf andere multimodale Aufgabenstellungen übertragbar ist.

## **8.2 Ausblick**

Die entwickelte multimodale Architektur etabliert eine methodische Grundlage für weiterführende Forschungsarbeiten in der automobilen KI-Domäne. Drei zentrale Entwicklungsrichtungen zeigen substanzielles Optimierungspotential für echtzeitfähige, präventive Fahrerassistenzsysteme.

### **Wahrnehmungsoptimierung**

Die YOLO-basierte Objekterkennung stellt den primären Engpass der Echtzeitverarbeitung dar. Eine zukünftige Entwicklung einer domänenspezifischen

Wahrnehmungsarchitektur für Autobahnszenarien verspricht signifikante Latenzreduktionen. Durch Ausnutzung der strukturierten Umgebungscharakteristika (perspektivische Geometrie, limitierte Objektklassen) können spezialisierte neuronale Netze mit reduzierter Parameteranzahl implementiert werden. Die Integration von Temporal Consistency Constraints durch Multi-Frame-Fusion würde zusätzlich die Robustheit bei gleichzeitiger Rechenoptimierung erhöhen.

### **Architekturerweiterung**

Die entwickelte modulare Architektur ermöglicht die systematische Integration und Evaluation zusätzlicher Encoder- und Fusion-Module ohne strukturelle Modifikationen der Gesamtarchitektur. Cross-Modal-Attention und Object-Query-Fusion sind bereits implementiert, konnten jedoch aufgrund numerischer Instabilitäten nicht in die vorliegende Evaluation einbezogen werden. Die modulare Struktur erlaubt zukünftigen Arbeiten die nahtlose Entwicklung weiterer Architekturvarianten und deren kontrollierte Bewertung im bestehenden Evaluationsframework.

### **Sensor-Fusion**

Durch die Integration zusätzlicher Sensordatenströme kann man in Zukunft die multimodale Wahrnehmung substanzial erweitern. LiDAR-Daten ermöglichen präzise Distanzmessungen und 3D-Objektlokalisierung, während Radar-Sensoren Geschwindigkeitsinformationen und Wetterabhängigkeit bieten. Vision Language Models (VLMs) können Verkehrsschilder und urbane Kontextinformationen semantisch interpretieren. Die Implementierung einer hierarchischen Fusion-Architektur mit sensortypspezifischen Encodern und einem zentralen Multi-Modal-Transformer würde diese Datenströme optimal integrieren.

# Literaturverzeichnis

- [1] A. Karpathy und A. Elluswamy, „Tesla AI Day 2021“, San Francisco, Kalifornien, 20. August 2021. [Online]. Verfügbar unter: <https://www.youtube.com/watch?v=j0z4FweCy4M&t=10755s>
- [2] K. Jetto, Z. Tahiri, A. Benyoussef, und A. El Kenz, „Cognitive anticipation cellular automata model: An attempt to understand the relation between the traffic states and rear-end collisions“, *Accid. Anal. Prev.*, Bd. 142, S. 105507, Juli 2020, doi: 10.1016/j.aap.2020.105507.
- [3] V. Mahajan, C. Katrakazas, und C. Antoniou, „Crash Risk Estimation Due to Lane Changing: A Data-Driven Approach Using Naturalistic Data“, *IEEE Trans. Intell. Transp. Syst.*, Bd. 23, Nr. 4, S. 3756–3765, Apr. 2022, doi: 10.1109/TITS.2020.3042097.
- [4] Statistisches Bundesamt, „Straßenverkehrsunfälle nach Unfallkategorie, Ortslage“. Zugriffen: 13. Juni 2025. [Online]. Verfügbar unter: <https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Tabellen/polizeilich-erfasste-unfaelle.html>
- [5] G. Johansson und K. Rumar, „Drivers' Brake Reaction Times“, *Hum. Factors J. Hum. Factors Ergon. Soc.*, Bd. 13, Nr. 1, S. 23–27, Feb. 1971, doi: 10.1177/001872087101300104.
- [6] H. Summala, „Brake Reaction Times and Driver Behavior Analysis“, *Transp. Hum. Factors*, Bd. 2, Nr. 3, S. 217–226, Sep. 2000, doi: 10.1207/STHF0203\_2.
- [7] G. F. Zhao, S. S. Wang, N. Yang, K. Qing, und M. M. Liu, „Study on Driver's Reaction Time of Emergency Braking“, *Adv. Mater. Res.*, Bd. 779–780, S. 925–928, Sep. 2013, doi: 10.4028/www.scientific.net/AMR.779-780.925.
- [8] D. Vignarca, S. Arrigoni, M. Vignati, und E. Sabbioni, „Analysis of Communication Delays in Roadside Detection Systems for Cooperative AEB Implementation“, in *2023 IEEE Vehicle Power and Propulsion Conference (VPPC)*, Milan, Italy: IEEE, Okt. 2023, S. 1–6. doi: 10.1109/VPPC60535.2023.10403218.
- [9] S. Behere und M. Törngren, „A functional reference architecture for autonomous driving“, *Inf. Softw. Technol.*, Bd. 73, S. 136–150, Mai 2016, doi: 10.1016/j.infsof.2015.12.008.
- [10] M. A. Manzour, C. M. Elias, E. I. Morgan, und O. M. Shehata, „Development of a Modular ROS-Enabled Pedestrian Intention Prediction Architecture for AVs Maneuvering Control“, *IEEE Trans. Intell. Transp. Syst.*, Bd. 26, Nr. 1, S. 798–809, Jan. 2025, doi: 10.1109/TITS.2024.3491972.
- [11] R. McAllister, B. Wulfe, J. Mercat, L. Ellis, S. Levine, und A. Gaidon, „Control-Aware Prediction Objectives for Autonomous Driving“, 2022, arXiv. doi: 10.48550/ARXIV.2204.13319.
- [12] J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang, und Z. Xiong, „Enhanced Object Detection With Deep Convolutional Neural Networks for Advanced Driving Assistance“, *IEEE Trans. Intell. Transp. Syst.*, Bd. 21, Nr. 4, S. 1572–1583, Apr. 2020, doi: 10.1109/TITS.2019.2910643.
- [13] M. Shirpour, N. Khairdoost, M. A. Bauer, und S. S. Beauchemin, „Traffic Object Detection and Recognition Based on the Attentional Visual Field of Drivers“, *IEEE Trans. Intell. Veh.*, Bd. 8, Nr. 1, S. 594–604, Jan. 2023, doi: 10.1109/TIV.2021.3133849.
- [14] Z. Sheng, Z. Huang, und S. Chen, „Kinematics-Aware Multigraph Attention Network with Residual Learning for Heterogeneous Trajectory Prediction“, *J. Intell. Connect. Veh.*, Bd. 7, Nr. 2, S. 138–150, Juni 2024, doi: 10.26599/JICV.2023.9210036.

- [15] Z. Yan, B. Yang, Z. Wang, und K. Nakano, „A Predictive Model of a Driver’s Target Trajectory Based on Estimated Driving Behaviors“, Sensors, Bd. 23, Nr. 3, S. 1405, Jan. 2023, doi: 10.3390/s23031405.
- [16] R. Shi und X. Wang, „Digitizing traffic rules to guide automated vehicle trajectory planning“, Expert Syst. Appl., Bd. 272, S. 126661, Mai 2025, doi: 10.1016/j.eswa.2025.126661.
- [17] T. Cai u. a., „Driving with Regulation: Interpretable Decision-Making for Autonomous Vehicles with Retrieval-Augmented Reasoning via LLM“, 2024, arXiv. doi: 10.48550/ARXIV.2410.04759.
- [18] „Tesla AI Day 2022“, gehalten auf der Tesla AI Day, 1. Oktober 2022. Zugriffen: 13. Juni 2025. [Online]. Verfügbar unter: [https://www.youtube.com/watch?v=ODSJsviD\\_SU](https://www.youtube.com/watch?v=ODSJsviD_SU)
- [19] Z. Huang, H. Liu, J. Wu, und C. Lv, „Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving“, IEEE Trans. Neural Netw. Learn. Syst., Bd. 35, Nr. 11, S. 15222–15236, Nov. 2024, doi: 10.1109/TNNLS.2023.3283542.
- [20] S. Luo, X. Li, und Z. Sun, „An Optimization-based Motion Planning Method for Autonomous Driving Vehicle“, in 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China: IEEE, Nov. 2020, S. 739–744. doi: 10.1109/ICUS50048.2020.9275009.
- [21] F. Barez, H. Hasanbieg, und A. Abbate, „System III: Learning with Domain Knowledge for Safety Constraints“, 2023, arXiv. doi: 10.48550/ARXIV.2304.11593.
- [22] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, und H. Li, „End-to-End Autonomous Driving: Challenges and Frontiers“, IEEE Trans. Pattern Anal. Mach. Intell., Bd. 46, Nr. 12, S. 10164–10183, Dez. 2024, doi: 10.1109/TPAMI.2024.3435937.
- [23] H.-H. Jebamikyous und R. Kashef, „Autonomous Vehicles Perception (AVP) Using Deep Learning: Modeling, Assessment, and Challenges“, IEEE Access, Bd. 10, S. 10523–10535, 2022, doi: 10.1109/ACCESS.2022.3144407.
- [24] L.-H. Wen und K.-H. Jo, „Deep learning-based perception systems for autonomous driving: A comprehensive survey“, Neurocomputing, Bd. 489, S. 255–270, Juni 2022, doi: 10.1016/j.neucom.2021.08.155.
- [25] A. Gupta, A. Anpalagan, L. Guan, und A. S. Khwaja, „Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues“, Array, Bd. 10, S. 100057, Juli 2021, doi: 10.1016/j.array.2021.100057.
- [26] R. Girshick, „Fast R-CNN“, 2015, arXiv. doi: 10.48550/ARXIV.1504.08083.
- [27] W. Zou, Z. Zhang, Y. Peng, C. Xiang, S. Tian, und L. Zhang, „SC-RPN: A Strong Correlation Learning Framework for Region Proposal“, IEEE Trans. Image Process., Bd. 30, S. 4084–4098, 2021, doi: 10.1109/TIP.2021.3069547.
- [28] X. Xie, G. Cheng, J. Wang, X. Yao, und J. Han, „Oriented R-CNN for Object Detection“, in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada: IEEE, Okt. 2021, S. 3500–3509. doi: 10.1109/ICCV48922.2021.00350.
- [29] Ultralytics, „Two-Stage Object Detectors“. [Online]. Verfügbar unter: <https://www.ultralytics.com/glossary/two-stage-object-detectors>
- [30] D. Oh u. a., „Low-Cost Object Detection Models for Traffic Control Devices through Domain Adaption of Geographical Regions“, Remote Sens., Bd. 15, Nr. 10, S. 2584, Mai 2023, doi: 10.3390/rs15102584.

- [31] H. Zhang, G. Luo, J. Li, und F.-Y. Wang, „C2FDA: Coarse-to-Fine Domain Adaptation for Traffic Object Detection“, IEEE Trans. Intell. Transp. Syst., Bd. 23, Nr. 8, S. 12633–12647, Aug. 2022, doi: 10.1109/TITS.2021.3115823.
- [32] X. Yu und X. Lu, „Domain Adaptation of Anchor-Free object detection for urban traffic“, Neurocomputing, Bd. 582, S. 127477, Mai 2024, doi: 10.1016/j.neucom.2024.127477.
- [33] H. Wang und H. Qian, „SDG-YOLOv8: Single-domain generalized object detection based on domain diversity in traffic road scenes“, Displays, Bd. 87, S. 102948, Apr. 2025, doi: 10.1016/j.displa.2024.102948.
- [34] W. Fang, W. Zhuo, J. Yan, Y. Song, D. Jiang, und T. Zhou, „Attention meets long short-term memory: A deep learning network for traffic flow forecasting“, Phys. Stat. Mech. Its Appl., Bd. 587, S. 126485, Feb. 2022, doi: 10.1016/j.physa.2021.126485.
- [35] Y. Tao, S. Hu, Z. Fang, und Y. Fang, „Directed-CP: Directed Collaborative Perception for Connected and Autonomous Vehicles via Proactive Attention“, 9. Mai 2025, arXiv: arXiv:2409.08840. doi: 10.48550/arXiv.2409.08840.
- [36] J. Li u. a., „Learning for Vehicle-to-Vehicle Cooperative Perception Under Lossy Communication“, IEEE Trans. Intell. Veh., Bd. 8, Nr. 4, S. 2650–2660, Apr. 2023, doi: 10.1109/TIV.2023.3260040.
- [37] S. Hochreiter und J. Schmidhuber, „Long Short-Term Memory“, Neural Comput., Bd. 9, Nr. 8, S. 1735–1780, Nov. 1997, doi: 10.1162/heco.1997.9.8.1735.
- [38] D. Wang, X. Liu, und J. Zhang, „Improved Vanishing Gradient Problem for Deep Multi-layer Neural Networks“, in Cognitive Systems and Information Processing, Bd. 1787, F. Sun, A. Cangelosi, J. Zhang, Y. Yu, H. Liu, und B. Fang, Hrsg., in Communications in Computer and Information Science, vol. 1787. , Singapore: Springer Nature Singapore, 2023, S. 159–173. doi: 10.1007/978-981-99-0617-8\_12.
- [39] M. Roodschild, J. Gotay Sardiñas, und A. Will, „A new approach for the vanishing gradient problem on sigmoid activation“, Prog. Artif. Intell., Bd. 9, Nr. 4, S. 351–360, Dez. 2020, doi: 10.1007/s13748-020-00218-y.
- [40] Safwan Mahmood Al-Selwi, Mohd Fadzil Hassan, Said Jadid Abdulkadir, und Amgad Muneer, „LSTM Inefficiency in Long-Term Dependencies Regression Problems“, J. Adv. Res. Appl. Sci. Eng. Technol., Bd. 30, Nr. 3, S. 16–31, Mai 2023, doi: 10.37934/araset.30.3.1631.
- [41] S.-H. Noh, „Analysis of Gradient Vanishing of RNNs and Performance Comparison“, Information, Bd. 12, Nr. 11, S. 442, Okt. 2021, doi: 10.3390/info12110442.
- [42] A. Vaswani u. a., „Attention Is All You Need“, 2017, arXiv. doi: 10.48550/ARXIV.1706.03762.
- [43] M.-I. Stan und O. Rhodes, „Learning long sequences in spiking neural networks“, Sci. Rep., Bd. 14, Nr. 1, S. 21957, Sep. 2024, doi: 10.1038/s41598-024-71678-8.
- [44] S. Reza, M. C. Ferreira, J. J. M. Machado, und J. M. R. S. Tavares, „A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks“, Expert Syst. Appl., Bd. 202, S. 117275, Sep. 2022, doi: 10.1016/j.eswa.2022.117275.
- [45] S. Islam u. a., „A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks“, 2023, arXiv. doi: 10.48550/ARXIV.2306.07303.
- [46] Y. Tay u. a., „Long Range Arena: A Benchmark for Efficient Transformers“, 8. November 2020, arXiv: arXiv:2011.04006. doi: 10.48550/arXiv.2011.04006.

- [47]Q. Fournier, G. M. Caron, und D. Aloise, „A Practical Survey on Faster and Lighter Transformers“, ACM Comput. Surv., Bd. 55, Nr. 14s, S. 1–40, Dez. 2023, doi: 10.1145/3586074.
- [48]K. Gadzicki, R. Khamsehashari, und C. Zetzsche, „Early vs Late Fusion in Multimodal Convolutional Neural Networks“, in 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa: IEEE, Juli 2020, S. 1–6. doi: 10.23919/FUSION45008.2020.9190246.
- [49]S. Y. Boulahia, A. Amamra, M. R. Madi, und S. Daikh, „Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition“, Mach. Vis. Appl., Bd. 32, Nr. 6, S. 121, Nov. 2021, doi: 10.1007/s00138-021-01249-8.
- [50]D. Vignarca, S. Arrigoni, M. Vignati, und E. Sabbioni, „Analysis of Communication Delays in Roadside Detection Systems for Cooperative AEB Implementation“, in 2023 IEEE Vehicle Power and Propulsion Conference (VPPC), Milan, Italy: IEEE, Okt. 2023, S. 1–6. doi: 10.1109/VPPC60535.2023.10403218.
- [51]L. Banjanović-Mehmedović und A. Husaković, „Edge AI: Reshaping the Future of Edge Computing with Artificial Intelligence“, in BASIC TECHNOLOGIES AND MODELS FOR IMPLEMENTATION OF INDUSTRY 4.0, Academy of Sciences and Arts of Bosnia and Herzegovina, Okt. 2023, S. 133–160. doi: 10.5644/PI2023.209.07.
- [52]B. Rokh, A. Azarpeyvand, und A. Khanteymoori, „A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification“, ACM Trans. Intell. Syst. Technol., Bd. 14, Nr. 6, S. 1–50, Dez. 2023, doi: 10.1145/3623402.
- [53]E. Jeong, J. Kim, und S. Ha, „TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards“, ACM Trans. Embed. Comput. Syst., Bd. 21, Nr. 5, S. 1–26, Sep. 2022, doi: 10.1145/3508391.
- [54]N. Galoppo, „Optimize AI pipelines with SYCL and OpenVINO“, in International Workshop on OpenCL, Bristol, United Kingdom United Kingdom: ACM, Mai 2022, S. 1–1. doi: 10.1145/3529538.3529561.
- [55]K. Vinoth und S. P, „Lightweight object detection in low light: Pixel-wise depth refinement and TensorRT optimization“, Results Eng., Bd. 23, S. 102510, Sep. 2024, doi: 10.1016/j.rineng.2024.102510.
- [56]ISO 14230-4:2000 Road vehicles — Diagnostic systems — Keyword Protocol 2000, 2000.
- [57]ISO 15765-2:2016 Road vehicles — Diagnostic communication over Controller Area Network (DoCAN), April 2016.
- [58]Guy Weemaes, Hrsg., „Liste der OBD2-Codes VW e-Up“, GoingElectric. [Online]. Verfügbar unter: <https://www.goingelectric.de/wiki/Liste-der-OBD2-Codes/>
- [59]buick301, „Mode 22 PID“, Freematics Forum. [Online]. Verfügbar unter: <https://forum.freemantics.com/viewtopic.php?t=2455>
- [60]Ford\_Dorf, „Extended CAN PID Codes Mode 22- OBD2“, Ford Employee Forum. [Online]. Verfügbar unter: <https://blueovalforums.com/forums/index.php?/topic/66993-extended-can-pid-codes-mode-22-obd2/>
- [61]Grant B, „Anatomy of a mode 22 OBD2 request?“, MX-5 Miata Forum. [Online]. Verfügbar unter: <https://forum.miata.net/vb/showthread.php?t=695208>
- [62]Brendan Whitfield, python-obd. [Online]. Verfügbar unter: <https://python-obd.readthedocs.io/en/latest/Custom%20Commands/>

- [63] Ultralytics, „YOLO trained on COCO“, Ultralytics Documentation. Zugegriffen: 14. Juni 2025. [Online]. Verfügbar unter: <https://docs.ultralytics.com/de/datasets/detect/coco>
- [64] T.-Y. Lin u. a., „Microsoft COCO: Common Objects in Context“, 2014, arXiv. doi: 10.48550/ARXIV.1405.0312.
- [65] Motional, „nulimages“. Zugegriffen: 13. Juni 2025. [Online]. Verfügbar unter: <https://www.nuscenes.org/nuimages#explore>
- [66] Karsten Behrendt, Boxy Vehicle Detection in Large Images. Kalifornien, USA, 2019. [Online]. Verfügbar unter: <https://boxy-dataset.com/boxy/>
- [67] Ultralytics, „TrainModel Training with Ultralytics YOLO“, YOLO Documentation. Zugegriffen: 14. Juni 2025. [Online]. Verfügbar unter: <https://docs.ultralytics.com/modes/train>
- [68] B. Singh und L. S. Davis, „An Analysis of Scale Invariance in Object Detection - SNIP“, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT: IEEE, Juni 2018, S. 3578–3587. doi: 10.1109/CVPR.2018.00377.
- [69] S. Li, J. Chen, W. Peng, X. Shi, und W. Bu, „A vehicle detection method based on disparity segmentation“, Multimed. Tools Appl., Bd. 82, Nr. 13, S. 19643–19655, Mai 2023, doi: 10.1007/s11042-023-14360-x.
- [70] B.-X. Wu, V. M. Shivanna, H.-H. Hung, und J.-I. Guo, „ConcentrateNet: Multi-Scale Object Detection Model for Advanced Driving Assistance System Using Real-Time Distant Region Locating Technique“, Sensors, Bd. 22, Nr. 19, S. 7371, Sep. 2022, doi: 10.3390/s22197371.
- [71] Ultralytics, Hrsg., „Object Detection“, Ultralytics YOLO Documentation. [Online]. Verfügbar unter: <https://docs.ultralytics.com/de/tasks/detect/>
- [72] Associate Professor, Mehr Chand Mahajan DAV College for Women, Chandigarh, India und I. Arora, „Improving Performance of Data Science Applications in Python“, Indian J. Sci. Technol., Bd. 17, Nr. 24, S. 2499–2507, Juni 2024, doi: 10.17485/IJST/v17i24.914.
- [73] „Breakdown: How Tesla will transition from Modular to End-To-End Deep Learning“, Welcome to The Library! Zugegriffen: 13. Juni 2025. [Online]. Verfügbar unter: <https://www.thinkautonomous.ai/blog/tesla-end-to-end-deep-learning/>
- [74] PyTorch, Hrsg., „Modules“. [Online]. Verfügbar unter: <https://docs.pytorch.org/docs/stable/notes/modules.html>
- [75] PyTorch, Hrsg., „LSTM“, PyTorch Dokumentation. [Online]. Verfügbar unter: <https://docs.pytorch.org/docs/stable/generated/torch.nn.LSTM.html>
- [76] PyTorch, Hrsg., „TransformerEncoder“, PyTorch Dokumentation. [Online]. Verfügbar unter: <https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html>
- [77] P. Thölke u. a., „Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data“, NeuroImage, Bd. 277, S. 120253, Aug. 2023, doi: 10.1016/j.neuroimage.2023.120253.
- [78] J. T. Hancock, T. M. Khoshgoftaar, und J. M. Johnson, „Evaluating classifier performance with highly imbalanced Big Data“, J. Big Data, Bd. 10, Nr. 1, S. 42, Apr. 2023, doi: 10.1186/s40537-023-00724-5.
- [79] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevey, und R. A. Bauder, „Investigating class rarity in big data“, J. Big Data, Bd. 7, Nr. 1, S. 23, Dez. 2020, doi: 10.1186/s40537-020-00301-0.
- [80] T.-Y. Lin, P. Goyal, R. Girshick, K. He, und P. Dollár, „Focal Loss for Dense Object Detection“, 2017, arXiv. doi: 10.48550/ARXIV.1708.02002.

- [81]J. Ansel u. a., „PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation“, in Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, La Jolla CA USA: ACM, Apr. 2024, S. 929–947. doi: 10.1145/3620665.3640366.
- [82]PyTorch, Hrsg., „Introduction to torch.compile“, PyTorch Dokumentation. [Online]. Verfügbar unter: [https://docs.pytorch.org/tutorials/intermediate/torch\\_compile\\_tutorial.html](https://docs.pytorch.org/tutorials/intermediate/torch_compile_tutorial.html)
- [83]W. Li, H. Liu, R. Ding, M. Liu, P. Wang, und W. Yang, „Exploiting Temporal Contexts With Strided Transformer for 3D Human Pose Estimation“, IEEE Trans. Multimed., Bd. 25, S. 1282–1293, 2023, doi: 10.1109/TMM.2022.3141231.
- [84]ARM Developer, Hrsg., „Arm Cortex-A Processor Comparison Table“.
- [85]Autopilot Review, „Tesla Hardware 4 (AI4)“, Autopilot Review. [Online]. Verfügbar unter: <https://www.autopilotreview.com/tesla-hardware-4-rolling-out-to-new-vehicles/>
- [86]Naush Patuck, „Introducing the Raspberry Pi AI HAT+ with up to 26 TOPS“. [Online]. Verfügbar unter: <https://www.raspberrypi.com/news/raspberry-pi-ai-hat/>