

**KI-gestützter Prototyp zur automatisierten Erkennung und Interpretation von  
Symbolen und Daten in PDF-basierten Gleisplänen**

**Masterarbeit**

**Utkarsh Swain**

**Matrikel-Nr.: 542847**

Gutachter: Prof. Dr.-Ing. David Inkermann (Institut für Maschinenwesen)  
2. Gutachter: Prof. Dr.-Ing. Armin Lohrengel (Institut für Maschinenwesen)  
Betreuer: M. Sc. Thomas Schumacher (Institut für Maschinenwesen)  
Betreuer in der Firma: Dipl.-Ing. Tobias Wolff (Siemens Mobility GmbH)

**Institut für Maschinenwesen  
Technische Universität Clausthal**

**13. Februar 2026**

## **Sperrvermerk Masterarbeit**

Die vorliegende Masterarbeit mit dem Titel

*„KI-gestützter Prototyp zur automatisierten Erkennung und Interpretation von Symbolen und Daten in PDF-basierten Gleisplänen“*

enthält interne vertrauliche Daten der Siemens Mobility GmbH.

Sie ist nur den Erst- und Zweitgutachtern sowie ggf. dem Prüfungsausschussvorsitzenden des Fachbereiches Maschinenbau zugänglich zu machen.

Veröffentlichungen und Vervielfältigungen der Masterarbeit, oder die Weitergabe der Masterarbeit – im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften – auch in digitaler Form – sind grundsätzlich untersagt.

Ausnahmen bedürfen der vorherigen schriftlichen Genehmigung der Siemens Mobility GmbH.

## **Eidesstattliche Erklärung**

Hiermit erkläre ich an Eides Statt, dass ich die bei der Technischen Universität Clausthal eingereichte Forschungsarbeit selbstständig und ohne unerlaubte Hilfe angefertigt habe. Die benutzten Hilfsmittel sind vollständig angegeben.

---

Datum und Unterschrift

Des Weiteren erkläre ich mich **nicht** damit einverstanden, dass meine Forschungsarbeit in der Instituts- und/oder der Universitätsbibliothek ausgelegt und aufbewahrt werden darf.

---

Datum und Unterschrift

# Inhaltsverzeichnis

<b>Formelzeichen- und Abkürzungsverzeichnis</b>	<b>xii</b>
<b>1 Abstract</b>	<b>xviii</b>
<b>2 Einleitung</b>	<b>1</b>
2.1 Motivation . . . . .	1
2.2 Problemstellung . . . . .	2
2.3 Zielsetzung der Arbeit . . . . .	4
2.4 Forschungsfragen . . . . .	5
2.5 Forschungsvorgehen . . . . .	6
2.6 Aufbau der Arbeit . . . . .	6
<b>3 Theoretische und technische Grundlagen</b>	<b>8</b>
3.1 Grafische Gleispläne im Bahnkontext . . . . .	8
3.1.1 Grundlagen des Gleisplans . . . . .	8
3.1.2 Bahntechnische Symbole und ihre Klassifikation . . . . .	9
3.1.3 Objekterkennung in technischen CAD-Zeichnungen . . . . .	11
3.1.4 Entwicklung der Objekterkennung . . . . .	12
3.1.5 Architekturen für Deep-Learning-basierte Objekterkennung . . . . .	12
3.1.6 Modelltraining . . . . .	14
3.1.7 Evaluationsmetriken . . . . .	15
3.2 Oriented Object Detection für rotierte Objekte . . . . .	17
3.2.1 Limitierungen achsenparalleler Bounding Boxes . . . . .	17
3.2.2 Oriented Bounding Boxes als Lösung . . . . .	19
3.2.3 Herausforderungen bei der Verwendung von OBB . . . . .	19
3.2.4 YOLOv8-OBB Architektur . . . . .	21
3.2.5 Relevanz für Gleisplananalyse . . . . .	22
3.3 Optical Character Recognition (OCR) . . . . .	23
3.3.1 Grundlagen und Entwicklung der OCR-Technologie . . . . .	23
3.3.2 Architekturen moderner OCR-Systeme . . . . .	24
3.3.3 OCR-Systeme im praktischen Einsatz . . . . .	26
3.3.4 Evaluationsmetriken für Texterkennung . . . . .	28
3.3.5 Herausforderungen bei technischen Zeichnungen . . . . .	28
3.3.6 Qualitätssicherung und Nachbearbeitung . . . . .	30
3.3.7 ROI-basierte OCR-Strategien . . . . .	32
3.3.8 Relevanz für die Gleisplananalyse . . . . .	33
3.4 Räumliche Beziehungen in technischen Dokumenten . . . . .	34

3.4.1	Das Prinzip der räumlichen Nähe . . . . .	34
3.4.2	Analogie zur Stücklistenzuordnung . . . . .	35
3.4.3	Herausforderung: Rotierte Elemente . . . . .	35
3.4.4	Lösung: Lokale Koordinatensysteme . . . . .	36
3.4.5	Klassenspezifische Suchstrategien . . . . .	36
3.5	Änderungserkennung zwischen Dokumentversionen . . . . .	37
3.5.1	Motivation: Das Revisionswesen . . . . .	37
3.5.2	Analogie: Stücklistenvergleich . . . . .	37
3.5.3	Kategorisierung von Änderungen . . . . .	38
3.5.4	Prinzip: Identifikation durch eindeutige Merkmale . . . . .	38
3.5.5	Das Zuordnungsproblem und der Hungarian Algorithm . . . . .	39
3.6	Konfigurationsbasierte Systemanpassung . . . . .	40
3.6.1	Das Problem der Domänenpezifität . . . . .	40
3.6.2	Lösung: Trennung von Code und Konfiguration . . . . .	41
3.6.3	Typische Konfigurationsparameter . . . . .	42
3.6.4	Klassenspezifische Parameter . . . . .	42
3.6.5	Orientierungsabhängige Verarbeitung . . . . .	42
3.7	Rückverfolgbarkeit und Qualitätssicherung . . . . .	43
3.7.1	Definition und Bedeutung . . . . .	43
3.7.2	Komponenten der Rückverfolgbarkeit . . . . .	43
3.7.3	Qualitätssicherung durch Validierung . . . . .	44
3.7.4	Umgang mit unsicheren Ergebnissen . . . . .	44
3.7.5	Human-in-the-Loop: Der Mensch als finale Instanz . . . . .	44
3.7.6	Zusammenfassung: Theoretische Grundlagen . . . . .	45
<b>4</b>	<b>Anforderungsanalyse</b> . . . . .	<b>47</b>
4.1	Funktionale Anforderungen . . . . .	47
4.1.1	Symbolerkennung und Objektklassifizierung . . . . .	47
4.1.2	Texterkennung und OCR-Integration . . . . .	49
4.1.3	Semantische Verknüpfung und Logik . . . . .	49
4.1.4	Datenaufbereitung und Export . . . . .	49
4.1.5	Benutzerinteraktion und Konfiguration . . . . .	50
4.2	Nicht-funktionale Anforderungen . . . . .	51
4.2.1	Sicherheit und Datenschutz . . . . .	51
4.2.2	Qualität und Zuverlässigkeit . . . . .	51
4.2.3	Effizienz und Wirtschaftlichkeit . . . . .	52
4.2.4	Wartbarkeit und Erweiterbarkeit . . . . .	52
4.2.5	Datenformate und Schnittstellen . . . . .	52
4.3	Herausforderungen bei der Umsetzung . . . . .	53
4.4	Anforderungs-Rückverfolgbarkeit . . . . .	53

---

<b>5 Konzeption des Prototyps</b>	<b>54</b>
5.1 Prototyparchitektur . . . . .	54
5.1.1 Dateneingabe . . . . .	54
5.1.2 Vorverarbeitung . . . . .	56
5.1.3 Detection & OCR . . . . .	57
5.1.4 Nachbearbeitung und Mapping . . . . .	57
5.1.5 Speicherung . . . . .	58
5.1.6 Benutzeroberfläche . . . . .	58
5.1.7 Export . . . . .	59
5.2 Workflow: Von PDF zu Excel . . . . .	59
5.2.1 1. Ebene - UI/Frontend (Benutzeroberfläche) . . . . .	59
5.2.2 2. Ebene - Backend/Verarbeitung (Kernlogik) . . . . .	60
5.2.3 3. Ebene - Speicher und Export (Datenverwaltung) . . . . .	61
5.3 Designentscheidungen . . . . .	61
5.3.1 Datenaufbereitung: PDF-Rasterisierung mit pdf2image . . . . .	62
5.3.2 Objekterkennung: Einsatz von YOLOv8 OBB . . . . .	63
5.3.3 Texterkennung: Multi-Engine-Strategie und Dual-Angle-Routing . . . . .	65
5.3.4 Datenexport und Reporting: Excel Schnittstelle . . . . .	67
5.3.5 Rückverfolgbarkeit: Koordinaten + Bildausschnitte . . . . .	68
5.3.6 Änderungsverfolgung: Konzeptioneller Ansatz . . . . .	69
5.3.7 Benutzeroberfläche: PyQt5 . . . . .	70
5.3.8 Datenpersistenz und Speicherschicht: PostgreSQL . . . . .	71
5.4 Linking- & Assoziationsmodul . . . . .	72
5.4.1 Konzeptioneller Ansatz . . . . .	73
5.4.2 Verknüpfungsstrategien . . . . .	73
5.5 Validierung und Qualitätssicherung . . . . .	73
<b>6 Implementierung</b>	<b>75</b>
6.1 Technologiestack und Entwicklungsumgebung . . . . .	75
6.1.1 Programmiersprache und Kernbibliotheken . . . . .	75
6.1.2 Entwicklungsumgebung . . . . .	76
6.1.3 Hardware-Infrastruktur . . . . .	76
6.1.4 Projektstruktur . . . . .	76
6.2 Objekterkennung mit YOLOv8-OBB . . . . .	76
6.2.1 Datensatzerstellung und Annotation . . . . .	77
6.2.2 Modelltraining und Optimierung . . . . .	83
6.2.3 Inferenz-Pipeline Implementierung . . . . .	89
6.3 Orientierungsadaptive OCR-Pipeline . . . . .	98
6.3.1 Multi-Engine Kaskadierung . . . . .	98
6.3.2 Dual-Winkel-Routing System . . . . .	100

6.3.3	Klassenspezifische Strategien . . . . .	103
6.3.4	Bildvorverarbeitungs-Algorithmen . . . . .	105
6.3.5	Validierung und Qualitätssicherung . . . . .	107
6.3.6	Zusammenfassung der Pipeline-Integration . . . . .	109
6.4	Intelligente Symbol-Text Verknüpfung . . . . .	110
6.4.1	Rotationsinvariante Koordinatentransformation . . . . .	111
6.4.2	Proximity-basierter Linking-Algorithmus . . . . .	112
6.4.3	Adaptive Learning Mechanismus . . . . .	115
6.4.4	Komplexe Verknüpfungslogik für Spezialfälle . . . . .	118
6.4.5	Zusammenfassung der Verknüpfungslogik . . . . .	120
6.5	Datenvalidierung und Qualitätssicherung . . . . .	121
6.5.1	Validierungsarchitektur . . . . .	121
6.5.2	Automatische Fehlerkorrektur . . . . .	124
6.5.3	Integration in die Benutzeroberfläche . . . . .	126
6.5.4	Persistierung und Export der Validierungsergebnisse . . . . .	129
6.5.5	Zusammenfassung . . . . .	131
6.6	Unterstützende Komponenten . . . . .	132
6.6.1	Versionsvergleich und Änderungsdetektion . . . . .	132
6.6.2	Rückverfolgbarkeit zur Quelldokumentation . . . . .	136
6.6.3	Persistente Datenhaltung . . . . .	137
6.7	Benutzeroberfläche . . . . .	142
6.7.1	Setup- und Analysedialog . . . . .	142
6.7.2	Architektur und Designentscheidungen . . . . .	143
6.7.3	PDF-Viewer mit Overlay-System . . . . .	143
6.7.4	Interaktive Ergebnistabelle . . . . .	145
6.7.5	Validierungs-Dialog . . . . .	146
6.7.6	Versionsvergleichs-Ansicht . . . . .	147
6.7.7	Export-Funktionalität . . . . .	149
6.8	Zusammenfassung . . . . .	154
6.8.1	Kernkomponenten der Pipeline . . . . .	154
6.8.2	Unterstützende Systeme . . . . .	155
6.8.3	Implementierungsumfang . . . . .	155
6.8.4	Deployment und Ausführung . . . . .	156
6.8.5	Schnittstellen und Erweiterbarkeit . . . . .	157
6.8.6	Abschließende Bemerkungen . . . . .	157
<b>7</b>	<b>Evaluation</b>	<b>158</b>
7.1	Testmethodik . . . . .	158
7.1.1	Testdatensätze . . . . .	158
7.1.2	Evaluationsmetriken . . . . .	160

---

7.1.3	Testumgebung . . . . .	162
7.2	Ergebnisanalyse . . . . .	162
7.2.1	Objekterkennungsleistung . . . . .	162
7.2.2	End-to-End Systemevaluation auf dem Testsatz . . . . .	167
7.2.3	Validierung weiterer funktionaler Anforderungen . . . . .	179
7.2.4	Validierung weiterer nicht-funktionaler Anforderungen . . . . .	180
7.2.5	Validierung der Export- und Hilfsfunktionen . . . . .	181
7.3	Anforderungs-Rückverfolgbarkeit . . . . .	182
7.3.1	Funktionale Anforderungen . . . . .	182
7.3.2	Nicht-funktionale Anforderungen . . . . .	184
7.4	Validierung aller Anforderungen . . . . .	185
7.5	Zusammenfassung der Evaluationsergebnisse . . . . .	185
<b>8</b>	<b>Diskussion und Ausblick</b> . . . . .	<b>188</b>
8.1	Einordnung der Ergebnisse . . . . .	188
8.1.1	Erfüllung der Kernziele . . . . .	188
8.1.2	Vergleich mit verwandten Arbeiten . . . . .	189
8.2	Kritische Reflexion . . . . .	190
8.2.1	Methodische Überlegungen . . . . .	190
8.2.2	Systemarchitektur und Design-Entscheidungen . . . . .	190
8.2.3	Nicht erreichte oder partiell erfüllte Anforderungen . . . . .	192
8.3	Identifizierte Limitationen . . . . .	193
8.3.1	Datenbezogene Limitationen . . . . .	193
8.3.2	Technische Limitationen . . . . .	194
8.3.3	Prozessuale Limitationen . . . . .	196
8.4	Verbesserungspotenziale . . . . .	197
8.4.1	Kurzfristige Optimierungen (0-6 Monate) . . . . .	197
8.4.2	Mittelfristige Erweiterungen (6-12 Monate) . . . . .	199
8.4.3	Langfristige Forschungsrichtungen (12+ Monate) . . . . .	201
8.5	Generalisierbarkeit und Übertragbarkeit . . . . .	204
8.5.1	Übertragbarkeit auf verwandte Domänen . . . . .	204
8.5.2	Generalisierbare Komponenten und Muster . . . . .	205
8.5.3	Nicht-übertragbare domänenspezifische Aspekte . . . . .	206
8.6	Ausblick und zukünftige Entwicklungen . . . . .	207
8.6.1	Evolution des konkreten Systems . . . . .	207
8.6.2	Trends in der automatisierten Dokumentenverarbeitung . . . . .	209
8.6.3	Gesellschaftliche und ethische Implikationen . . . . .	211
8.7	Beantwortung der Forschungsfragen . . . . .	212
8.7.1	Hauptforschungsfrage (HF) . . . . .	212
8.7.2	Teilforschungsfrage 1 (TF1) . . . . .	213

8.7.3 Teilstudie 2 (TF2) . . . . .	213
8.7.4 Teilstudie 3 (TF3) . . . . .	214
8.7.5 Teilstudie 4 (TF4) . . . . .	214
8.7.6 Zusammenfassung . . . . .	215
8.8 Abschließende Bewertung . . . . .	215

# Abbildungsverzeichnis

3.1 Ausschnitt eines schematischen Gleisplans mit drei Bahnsteigbereichen (4a/4b, 1/2, 3), wobei Richtungspfeile die zulässigen Fahrtrichtungen kennzeichnen (Beispielhafte Darstellung) . . . . .	9
3.2 Architekturvergleich zwischen zweistufigen und einstufigen Objektdetektoren. . . . .	14
3.3 Visualisierung der Intersection over Union (IoU) bei unterschiedlichen Überlappungsgraden. . . . .	16
3.4 AABB vs OBB Vergleich bei rotiertem Textlabel. . . . .	18
3.5 NMS-Problem bei dicht platzierten rotierten Objekten. . . . .	19
3.6 Hauptherausforderungen bei OBB-Verwendung: IoU-Berechnung erfordert Polygon-Clipping statt einfacher Min/Max-Operationen (links), zyklische Winkel führen zu Gradienteninstabilität (Mitte), und der zusätzliche Rotationsparameter erhöht den Trainingsaufwand (rechts). . . . .	20
3.7 YOLOv8-OBB Netzwerkarchitektur (nach [22]). . . . .	21
3.8 CRNN-Architektur . . . . .	24
3.9 PaddleOCR-Pipeline: Dreistufige Architektur mit Detektion, Winkelkorrektur und Erkennung. . . . .	27
3.10 Zentrale Herausforderungen der OCR in technischen Zeichnungen. . . . .	28
3.11 Dual-Pfad ROI-Extraktion: Kardinale Orientierungen verwenden effiziente affine Rotation, während schräge Texte eine Perspektivtransformation erfordern. . . . .	32
3.12 Illustration des Gestaltprinzips der räumlichen Nähe . . . . .	34
3.13 Zuordnung von Positionsnummern zu Bauteilen in einer Explosionsdarstellung . .	35
3.14 Ambiguität von Richtungsbegriffen bei rotierten Symbolen . . . . .	35
3.15 Transformation vom globalen ins lokale Koordinatensystem eines Symbols . . .	36
3.16 Schematische Darstellung von Änderungen zwischen zwei Planversionen. . . . .	38
3.17 Variabilität von Symboldarstellungen und Bezeichnungskonventionen. . . . .	41
3.18 Architekturprinzip der Trennung von Programmlogik und Konfiguration. . . . .	41
3.19 Struktur eines rückverfolgbaren Datensatzes mit Quellenmetadaten. . . . .	43
3.20 Konfidenzbasierte Klassifikation von Erkennungsergebnissen . . . . .	44
4.1 Beispielhafte Extraktion von Text und Position am Symbol GKS . . . . .	49
4.2 Schematische Übertragung von erkannten Objektdaten in die Ziel-Tabelle . . . .	50
4.3 Visualisierung der Änderungsverfolgung zwischen zwei Planversionen . . . . .	50
4.4 Visuelle Validierung durch Bounding-Box-Overlays im Gleisplan [4] . . . . .	50
5.1 Modulare Prototyparchitektur des Gleisplanextraktors . . . . .	54
5.2 Differenzierte Eingabeverarbeitung für PDF- und Bilddateien. . . . .	55
5.3 Workflowdiagramm des Prototyps . . . . .	59

5.4 Konzeptioneller Vergleich der Dual-Winkel-Verarbeitungsstrategien: Cardinal Path für nahezu ausgerichtete Texte ( $ \theta  \leq 15^\circ$ ) mit diskreter Rotation; Angular Path für stark geneigte Texte ( $ \theta  > 15^\circ$ ) mit Perspektiventransformation . . . . .	66
6.1 COCO-konforme Datensetstruktur für das YOLOv8-OBB-Training . . . . .	81
6.2 Charakteristika des Trainingsdatensatzes (923 Bilder, 12.506 Instanzen): (oben links) Klassenverteilung mit absoluten Instanzzahlen, (unten links) räumliche Verteilung der Bounding-Box-Zentren, (unten rechts) Größenverteilung der annotierten Objekte in normalisierten Koordinaten. . . . .	82
6.3 Trainings- und Validierungsverläufe über 120 Epochen: (oben) Trainings-Losses und Metriken, (unten) Validierungs-Losses und mAP-Werte. . . . .	85
6.4 Precision-Recall-Kurven für alle 13 Symbolklassen auf dem Validierungsdatensatz. Die Werte in der Legende geben die jeweilige AP@0.5 an. . . . .	87
6.5 Normalisierte Konfusionsmatrix auf dem Validierungsdatensatz. Zeilen repräsentieren die vorhergesagten Klassen, Spalten die tatsächlichen Klassen. . . . .	88
6.6 F1-Score in Abhängigkeit der Konfidenzschwelle. Das globale Optimum liegt bei $F1 = 0,96$ bei einer Schwelle von 0,674. . . . .	89
6.7 Ablauf der Inferenz-Pipeline . . . . .	90
6.8 Formatabhängige Eingabeverarbeitung vor der einheitlichen Tiling-Pipeline. . . . .	91
6.9 2D-Tiling-Strategie mit quadratischen Kacheln . . . . .	92
6.10 Visualisierung der Halo-Strategie . . . . .	94
6.11 Adaptive Expansion der Region of Interest (ROI) . . . . .	96
6.12 Ablaufdiagramm der Multi-Engine OCR-Kaskadierung . . . . .	100
6.13 Vergleich: Cardinal Path vs. Angular Path Rotationsstrategien . . . . .	102
6.14 Verarbeitung eines um $37,5^\circ$ rotierten Elemente durch Angular-Path-OCR . . . . .	102
6.15 Morphologisches Opening zur Rauschentfernung bei GKS-Symbolen . . . . .	104
6.16 Schritte der Bildvorverarbeitung (schematische Darstellung) . . . . .	105
6.17 Gewichtung der Komponenten im Konfidenz-Scoring . . . . .	108
6.18 Prinzip der rotationsinvarianten Koordinatentransformation . . . . .	112
6.19 Probabilistische Fenstersuche basierend auf gelernten Offset-Patterns . . . . .	118
6.20 Geometrische Bestimmung der Fahrtrichtung eines Signals . . . . .	119
6.21 Dreistufige Validierungsarchitektur mit hierarchischer Fehleranalyse . . . . .	122
6.22 Konfidenzbasierte Entscheidungslogik für Korrekturvorschläge . . . . .	125
6.23 Schematische Darstellung der fünfteiligen Tab-Struktur des Validierungsdialogs .	127
6.24 Farbkodierung für Validierungsprobleme und Korrekturvorschläge . . . . .	127
6.25 Jump-to-Detection Workflow zwischen Validierungsdialog und Gleisplan-Ansicht	128
6.26 PostgreSQL Datenbankschema für Validierungspersistierung . . . . .	130
6.27 Schematische Darstellung des UID-Generierungsprozesses für verschiedene Objekttypen . . . . .	133
6.28 Mengentheoretische Visualisierung des Diff-Algorithmus . . . . .	134

6.29 Versionsvergleichs-Dialog mit tabellarischer Auflistung aller Änderungen zwischen zwei Planversionen . . . . .	135
6.30 GKS 0305 in Version 1: Koordinate 0.1770 . . . . .	136
6.31 GKS 0305 in Version 2: Koordinate 0.1729 . . . . .	136
6.32 Hierarchische Struktur der Metadaten für ein extrahiertes Objekt . . . . .	136
6.33 Datenbankschema mit Kardinalitäten (vereinfachte Darstellung) . . . . .	138
6.34 Setup- und Analyse-Dialog der Anwendung „RailDoc Studio“ . . . . .	142
6.35 Viewport-basiertes Culling zur Performance-Optimierung . . . . .	144
6.36 Hauptfenster mit PDF-Viewer (links), hierarchischer Ergebnistabelle (rechts) und bidirektonaler Navigation . . . . .	146
6.37 Änderungsvergleich-Dialog . . . . .	149
6.38 Export-Dialog mit klassenbasierter Auswahl, Live-Vorschau und Zielpositionierung für bestehende Dateien . . . . .	151
6.39 Exportierte Excel-Datei mit separaten Arbeitsblättern pro Objektklasse (sichtbar am unteren Rand: gks_festkodiert, gks_gesteuert, gm_block, etc.) . . . . .	152
6.40 Export-Dialog mit Formatauswahl. Das Dropdown-Menü (rechts) bietet drei Ausgabeformate: Excel (.xlsx), CSV (.csv) und JSON (.json). . . . .	153
6.41 Änderungsexport . . . . .	154
7.1 Precision-Recall-Kurven für alle Objektklassen auf dem Validierungsdatensatz. Die Fläche unter jeder Kurve entspricht dem klassenspezifischen AP-Wert. Der Gesamtwert mAP@0.5 beträgt 98,4 %. . . . .	164
7.2 Konfusionsmatrix der YOLO-Klassifikation (Validierungsset). Die Matrix zeigt, dass 4 Instanzen von <i>gks_gesteuert</i> fälschlicherweise als <i>gks_festkodiert</i> klassifiziert wurden. . . . .	166

# Tabellenverzeichnis

3.1 Klassifikation bahntechnischer Symbole nach funktionalen Kategorien . . . . .	10
3.2 Vergleich von CRNN- und Transformer-basierten OCR-Architekturen . . . . .	26
3.3 Vergleich etablierter OCR-Systeme für technische Anwendungen . . . . .	28
3.4 Regex-Validierungsmuster für Gleisplan-Textklassen . . . . .	31
3.5 Typische OCR-Homoglyphen und deren Korrekturmöglichkeiten . . . . .	31
3.6 Fuzzy-Matching mittels Levenshtein-Distanz . . . . .	31
3.7 Domänenspezifische Konventionen für Textpositionen relativ zu Symbolen . . . . .	37
3.8 Die drei Grundtypen von Änderungen zwischen Dokumentversionen . . . . .	38
3.9 Vergleich von Greedy- und Hungarian-Algorithmus für das Zuordnungsproblem .	40
3.10 Kategorien konfigurierbarer Parameter . . . . .	42
3.11 Wesentliche Metadaten für die Rückverfolgbarkeit . . . . .	44
4.1 13 Symbolklassen für die Datenextraktion . . . . .	49
4.2 Übersicht der unterstützten Datenformate (siehe 4.2.5 und 4.2.5) . . . . .	52
5.1 Vergleich verschiedener PDF-Verarbeitungsbibliotheken . . . . .	63
5.2 Vergleich und Bewertung von Alternativen zu YOLOv8 . . . . .	64
5.3 Vergleich und Bewertung alternativer OCR-Ansätze . . . . .	67
5.4 Vergleich der Export-Alternativen zu pandas + openpyxl . . . . .	68
5.5 Alternativen zur hybriden Rückverfolgbarkeit . . . . .	69
5.6 Alternative zu PyQt5 . . . . .	71
5.7 Alternativen zu PostgreSQL . . . . .	72
6.1 Übersicht der verwendeten Technologien und Bibliotheken . . . . .	75
6.2 Aufteilung des Datensatzes in Trainings- und Validierungssatz . . . . .	80
6.3 Verteilung der Annotationen über alle 13 Klassen im Trainings- und Validierungssatz. Die Spalte <i>Bilder</i> zeigt die Anzahl der Bilder mit mindestens einer Instanz der jeweiligen Klasse; da Bilder mehrere Klassen gleichzeitig enthalten, ist die Summe der Einzelwerte größer als die mit * markierte Gesamtbildzahl. . . . .	81
6.4 Hyperparameter des YOLOv8l-OBB-Trainings . . . . .	84
6.5 Aggregierte Detektionsmetriken auf dem Validierungsdatensatz (208 Bilder) . . .	86
6.6 Klassenspezifische AP@0.5 auf dem Validierungsdatensatz, sortiert nach absteigender AP@0.5 . . . . .	87
6.7 Winkeladaptive Bounding-Box-Parameter . . . . .	95
6.8 Klassenspezifische NMS-Schwellenwerte . . . . .	97
6.9 Klassenspezifische Padding-Faktoren und deren Begründung . . . . .	103
6.10 Validierungs-Patterns pro Symbolklasse mit Klassifizierung . . . . .	107

6.11 Klassenspezifische Linking-Parameter. Die Wertebereiche für $dy_{\max}$ und $dx_{\max}$ basieren auf typischen Symbolgrößen von 50–80 Pixeln bei 500 DPI. . . . .	113
6.12 Feste Suchparameter für Spezialfälle (kalibriert für 500 DPI) . . . . .	114
6.13 Beispielhafte Change Report Struktur mit verschiedenen Änderungstypen . . . . .	134
6.14 Codeumfang der Implementierungsmodule (gerundet) . . . . .	156
7.1 Charakteristika des Testdatensatzes (A0-Gleispläne) . . . . .	159
7.2 Komplexitätskategorien der A0-Testpläne . . . . .	160
7.3 Hardware- und Softwarekonfiguration der Testumgebung . . . . .	162
7.4 Aggregierte Detektionsmetriken auf dem Validierungsdatensatz . . . . .	163
7.5 Detektionsmetriken für alle 13 Klassen auf dem Validierungsdatensatz. mAP@0.5 aus den Precision-Recall-Kurven. . . . .	165
7.6 End-to-End Genauigkeit pro Objektklasse (9 Testpläne) . . . . .	167
7.7 End-to-End Systemgenauigkeit im Vergleich zum Anforderungsziel . . . . .	167
7.8 Attribut-spezifische Genauigkeit für Signale (9 Testpläne). Ein Signal gilt als vollständig korrekt, wenn alle drei Attribute fehlerfrei extrahiert wurden. . . . .	168
7.9 Rotationsanalyse . . . . .	168
7.10 Detaillierte End-to-End Ergebnisse pro Testplan mit Fehlertyp-Annotation. FP = False Positives (Extra-Erkennungen), die im Validierungsschritt entfernt werden können und daher nicht als Fehler gezählt werden. . . . .	169
7.11 Fehlerverteilung nach Plan mit Root Causes (9 Testpläne) . . . . .	170
7.12 Verteilung der End-to-End Fehler nach Fehlerursache (9 Pläne) . . . . .	171
7.13 End-to-End Accuracy nach Plankomplexität (alle 9 Testpläne) . . . . .	173
7.14 Zeitvergleich: Vollständige vs. gezielte Qualitätsprüfung mit Validierungswerkzeugen	175
7.15 Verarbeitungszeiten nach Plankomplexität (CPU-only, AMD Ryzen 5 PRO 5650U)	177
7.16 Zeitverteilung der Pipeline-Stufen nach Plankomplexität (Durchschnittswerte) .	177
7.17 Zeitvergleich: Manueller vs. KI-gestützter Prozess (Durchschnitt über 9 Testpläne)	178
7.18 Validierung funktionaler Anforderungen durch systematische Funktionstests .	180
7.19 Rückverfolgbarkeitsmatrix: Funktionale Anforderungen . . . . .	184
7.20 Rückverfolgbarkeitsmatrix: Nicht-funktionale Anforderungen . . . . .	184
7.21 Validierung aller funktionalen und nicht-funktionalen Anforderungen . . . . .	185

# Formelzeichen- und Abkürzungsverzeichnis

## Formelzeichen

$A$	Menge (z. B. Symbolmenge Version A)
$\alpha$	Skalierungsfaktor für Padding
$B$	Menge (z. B. Symbolmenge Version B) oder Bounding Box
$\beta$	Skalierungsfaktor für Padding
$B_{\text{gt}}$	Ground Truth Bounding Box
$B_{\text{pred}}$	Vorhergesagte Bounding Box
$c$	Konfidenzwert (Wahrscheinlichkeit)
$c_x, c_y$	Zentrumskoordinaten einer Bounding Box
$d$	Euklidische Distanz
$\vec{d}$	Distanzvektor
$d_k$	Dimensionalität der Key-Vektoren
$d_{\text{Lev}}$	Levenshtein-Distanz
$\delta$	Abweichungsschwellenwert
$\Delta$	Differenzmenge oder Versatz
$dx, dy$	Distanzkomponenten
$\epsilon$	Toleranzwert
$e_x, e_y$	Expansionsfaktoren für Bounding Boxes
$f$	Dateiname oder Feldbezeichner
$\gamma$	Normalisierte Konfidenz (OCR)
$h$	Höhe (Bounding Box oder Objekt)
$h_c$	Resultierende Zeichenhöhe in Pixeln
$\mathbf{H}$	Homographie-Matrix
$H$	Gesamthöhe des Bildes
$k$	Verhältnisfaktor (Schrifthöhe)
$k_{dx}, k_{dy}$	Klassenspezifische Multiplikatoren
$K$	Key-Matrix (Attention)

$\mathcal{L}$	Verlustfunktion (Loss)
$\lambda$	Gewichtungsfaktor in der Verlustfunktion
$\mu$	Erwartungswert / Mittelwert
$M$	Transformationsmatrix
$M_{\text{affin}}$	Affine Rotationsmatrix
$N$	Anzahl (z. B. Objekte, Klassen)
$O$	Überlappung (Overlap)
$p$	Padding
$P$	Wahrscheinlichkeit
$\pi$	Pfad im CTC-Algorithmus
$Q$	Query-Matrix (Attention)
$r$	Auflösung oder Radius
$R$	Rotationsmatrix
$s$	Schriftgröße oder Scrollposition
$S$	Schrittweite (Stride) oder Score
$\sigma$	Standardabweichung
$t$	Zeitindex oder Zeitdauer
$\tau$	Schwellenwert (Threshold)
$\theta$	Rotationswinkel
$t_x, t_y$	Translationskomponenten
$T$	Länge einer Sequenz oder Tile-Größe
$v$	Wert eines Datenfeldes
$V$	Value-Matrix (Attention)
$w$	Breite (Bounding Box oder Objekt)
$W$	Gesamtbreite des Bildes
$x, y$	Kartesische Koordinaten
$z$	Zoomfaktor
$\oplus$	Konkatenation
$\cap, \cup, \setminus$	Mengenoperationen

**Abkürzungen**

<b>AABB</b>	Axis-Aligned Bounding Box
<b>ACID</b>	Atomicity, Consistency, Isolation, Durability
<b>AGPL</b>	Affero General Public License
<b>AP</b>	Average Precision
<b>API</b>	Application Programming Interface
<b>ASCII</b>	American Standard Code for Information Interchange
<b>AWS</b>	Amazon Web Services
<b>BOM</b>	Bill of Materials
<b>BSD</b>	Berkeley Software Distribution
<b>BYTEA</b>	Byte Array
<b>CAD</b>	Computer-Aided Design
<b>CER</b>	Character Error Rate
<b>CLI</b>	Command Line Interface
<b>CNN</b>	Convolutional Neural Network
<b>COCO</b>	Common Objects in Context
<b>CPU</b>	Central Processing Unit
<b>CRNN</b>	Convolutional Recurrent Neural Network
<b>CSV</b>	Comma-Separated Values
<b>CTC</b>	Connectionist Temporal Classification
<b>CVAT</b>	Computer Vision Annotation Tool
<b>DB</b>	Differentiable Binarization
<b>DETR</b>	Detection Transformer
<b>DFL</b>	Distribution Focal Loss
<b>DIN</b>	Deutsches Institut für Normung
<b>DINOv2</b>	Self-supervised Vision Transformer
<b>DPI</b>	Dots Per Inch
<b>EN</b>	Europäische Norm
<b>ERP</b>	Enterprise Resource Planning

<b>FA</b>	Funktionale Anforderung
<b>FDA</b>	Food and Drug Administration
<b>FEM</b>	Finite Element Method
<b>FN</b>	False Negative
<b>FP</b>	False Positive
<b>FPN</b>	Feature Pyramid Network
<b>FPS</b>	Frames Per Second
<b>GIN</b>	Generalized Inverted Index
<b>GKS</b>	Gleiskoppelspule
<b>GM</b>	Gleismagnet
<b>GmbH</b>	Gesellschaft mit beschränkter Haftung
<b>GPU</b>	Graphics Processing Unit
<b>GUI</b>	Graphical User Interface
<b>HBB</b>	Horizontal Bounding Box
<b>HMM</b>	Hidden Markov Model
<b>HOG</b>	Histogram of Oriented Gradients
<b>ID</b>	Identifier
<b>IO</b>	Input/Output
<b>IoU</b>	Intersection over Union
<b>ISO</b>	International Organization for Standardization
<b>JPEG</b>	Joint Photographic Experts Group
<b>JPG</b>	Joint Photographic Experts Group
<b>JS</b>	JavaScript
<b>JSON</b>	JavaScript Object Notation
<b>JSONB</b>	JSON Binary
<b>KI</b>	Künstliche Intelligenz
<b>km</b>	Kilometer
<b>LCAD</b>	Legacy Computer-Aided Design
<b>LOC</b>	Lines of Code

<b>LSTM</b>	Long Short-Term Memory
<b>mAP</b>	mean Average Precision
<b>MIT</b>	Massachusetts Institute of Technology License
<b>ML</b>	Machine Learning
<b>MVC</b>	Model-View-Controller
<b>NASA-TLX</b>	NASA Task Load Index
<b>NFA</b>	Nicht-funktionale Anforderung
<b>NMS</b>	Non-Maximum Suppression
<b>OBB</b>	Oriented Bounding Box
<b>OCR</b>	Optical Character Recognition
<b>OOM</b>	Out of Memory
<b>OpenGL</b>	Open Graphics Library
<b>ORDBMS</b>	Object-Relational Database Management System
<b>ORM</b>	Object-Relational Mapping
<b>PANet</b>	Path Aggregation Network
<b>PDF</b>	Portable Document Format
<b>PIL</b>	Python Imaging Library
<b>PNG</b>	Portable Network Graphics
<b>PSM</b>	Page Segmentation Mode
<b>PyQt</b>	Python bindings for Qt
<b>QA</b>	Quality Assurance
<b>Randi</b>	Rohrleitungs- und Instrumentenfließschema
<b>RAM</b>	Random Access Memory
<b>RGB</b>	Red Green Blue
<b>RNN</b>	Recurrent Neural Network
<b>ROI</b>	Region of Interest
<b>RPN</b>	Region Proposal Network
<b>RTMDet</b>	Real-time Object Detection
<b>SGD</b>	Stochastic Gradient Descent

<b>SPS</b>	Speicherprogrammierbare Steuerung
<b>SQL</b>	Structured Query Language
<b>STN</b>	Spatial Transformer Network
<b>SUS</b>	System Usability Scale
<b>SVM</b>	Support Vector Machine
<b>TCP</b>	Tool Center Point
<b>TP</b>	True Positive
<b>UCS</b>	User Coordinate System
<b>UI</b>	User Interface
<b>UID</b>	Unique Identifier
<b>UPSERT</b>	Update or Insert
<b>UX</b>	User Experience
<b>VDI</b>	Verein Deutscher Ingenieure
<b>ViT</b>	Vision Transformer
<b>VRAM</b>	Video Random Access Memory
<b>WER</b>	Word Error Rate
<b>XLSX</b>	Excel Open XML Spreadsheet
<b>XML</b>	Extensible Markup Language
<b>XSS</b>	Cross-Site Scripting
<b>YOLO</b>	You Only Look Once

# 1. Abstract

Railway infrastructure planning relies heavily on technical layout documents in PDF or image format, which contain complex arrangements of symbols, text annotations, and positional information. Manual extraction and interpretation of this information is time-consuming, error-prone, and difficult to scale across projects and customers. This thesis presents the design and prototypical implementation of an intelligent, modular pipeline for automated analysis of railway track layouts (Gleispläne), developed in cooperation with Siemens Mobility GmbH. The system combines deep learning-based object detection using YOLOv8 with Oriented Bounding Boxes (OBB) for rotation-invariant symbol recognition, a multi-engine OCR cascade (PaddleOCR, Tesseract, EasyOCR) with orientation-adaptive preprocessing for robust text extraction, and an adaptive spatial linking algorithm that automatically associates detected symbols with their textual labels. Rule-based logic determines spatial relationships such as driving direction (Fahrtrichtung) and element groupings. All extracted objects and metadata are persisted in a PostgreSQL backend with JSONB storage, versioned records, and change tracking for full traceability between layout versions. The solution is evaluated on nine production A0 layouts from Siemens Mobility projects, achieving **97.22% end-to-end accuracy** for all 12 symbol classes, integrating detection, OCR, and spatial linking in a unified evaluation metric. The system reduces manual processing time by **89.7%**, from an average of 102 minutes to approximately 10.6 minutes per layout. A comprehensive PyQt5-based desktop application provides interactive visualization, manual correction tools, and validation overlays for human-in-the-loop quality assurance. This work contributes to the digitalization of railway infrastructure planning by enabling scalable, automated layout analysis that significantly reduces manual effort while improving data integrity and decision-making efficiency in safety-critical railway systems.

**Keywords:** Deep Learning, Object Detection, YOLOv8-OBB, Optical Character Recognition, Railway Infrastructure, Document Analysis, Automated Data Extraction

## 2. Einleitung

Die Digitalisierung revolutioniert zunehmend die traditionellen Arbeitsprozesse in der Industrie und eröffnet neue Möglichkeiten zur Steigerung der Effizienz, Verbesserung der Qualität und Senkung der Kosten. In vielen Bereichen des Ingenieurwesens, vom Maschinenbau über die Verfahrenstechnik bis hin zur Infrastrukturplanung, besteht ein signifikantes Potenzial für digitale Transformationsprozesse, insbesondere bei der Verarbeitung und Interpretation technischer Pläne und Zeichnungen. Die vorliegende Masterarbeit, die innerhalb der Siemens Mobility GmbH im Bereich der Bahninfrastruktur durchgeführt wurde, adressiert diese branchenübergreifende Herausforderung durch die Entwicklung eines KI-unterstützten Systems zur automatischen Verarbeitung technischer Pläne am konkreten Anwendungsfall der Gleispläne.

Im spezifischen Kontext der Schieneninfrastruktur bilden Stellwerke das Herzstück moderner Bahnsysteme und gewährleisten durch ihre komplexe Steuerungstechnik den sicheren und effizienten Zugverkehr. Die präzise Dokumentation ihrer Komponenten, von Signalen über Weichen bis hin zu Gleiskoppelpulsen, ist dabei von essentieller Bedeutung für Planung, Wartung und Modernisierung. Ähnlich wie bei Hydraulikschaltplänen im Maschinenbau oder R&I-Diagrammen in der Verfahrenstechnik erfolgt die Übertragung dieser Informationen von technischen Zeichnungen in strukturierte Datenformate jedoch weitgehend manuell, was sowohl zeit- als auch ressourcenintensiv ist [1].

Die fortschreitende Entwicklung in den Bereichen Computer Vision, Machine Learning und Prozessautomatisierung eröffnet jetzt weitere Perspektiven für die Optimierung dieser Arbeitsabläufe. Diese technologischen Möglichkeiten bilden die Grundlage für den in dieser Arbeit verfolgten Ansatz, der darauf abzielt, den Prozess der Datenextraktion und -verwaltung grundlegend zu modernisieren.

Ziel dieser Arbeit ist es, theoretische Konzepte der künstlichen Intelligenz mit den praktischen Anforderungen der technischen Planverarbeitung zu verknüpfen. Durch die Entwicklung und Validierung eines KI-gestützten Ansatzes am Beispiel der Schienenverkehrstechnik wird demonstriert, wie die Lücke zwischen analogen Planungsunterlagen und digitalen Datenmodellen geschlossen werden kann.

### 2.1 Motivation

Die Digitalisierung von technischen Planungsprozessen stellt eine zentrale Herausforderung für Unternehmen im Infrastruktursektor und im allgemeinen Maschinenbau dar. Der Übergang von papierbasierter oder halbdigitaler Dokumentation zu durchgängig digitalen Workflows verspricht erhebliche Effizienzsteigerungen und bildet die Grundlage für moderne Konzepte wie den Digitalen Zwilling (ein virtuelles Abbild physischer Systeme zur Simulation und Analyse) oder

modellbasierte Systementwicklung.

Im Bereich der Eisenbahnsignaltechnik beruhen viele Arbeitsschritte noch auf der manuellen Auswertung von Gleisplänen, in denen eine Vielzahl von Symbolen, Textbausteinen und kunden-individuellen Darstellungen enthalten ist. Neben dem hohen Arbeitsaufwand erschwert diese Vorgehensweise die systematische Versionierung und Rückverfolgbarkeit von Änderungen über den gesamten Projektlebenszyklus [2].

Ziel der vorliegenden Masterarbeit ist es daher, eine intelligente und automatisierte Lösung zu entwickeln, die durch den Einsatz moderner Deep-Learning-Verfahren eine zuverlässige Erkennung und Interpretation von Symbolen und Text in technischen Plänen ermöglicht. In Kombination mit regelbasierter Logik sowie strukturierten Ausgabeformaten entsteht ein skalierbarer Prototyp zur Verarbeitung, Auswertung und semantischen Interpretation von Gleisplänen.

Die Umsetzung einer solchen Lösung verspricht nicht nur eine erhebliche Effizienzsteigerung in der Planungs- und Prüfphase, sondern schafft auch eine fundierte Grundlage für Rückverfolgbarkeit, Änderungsmanagement und automatisierte Konsistenzprüfungen. Damit leistet die Arbeit einen praxisnahen Beitrag zur digitalen Transformation innerhalb der Siemens Mobility GmbH und zeigt exemplarisch auf, wie moderne Verfahren der Computer Vision konkret auf industrielle Anwendungsfälle im Ingenieurwesen übertragen werden können.

### Praxisbeispiel bei Siemens Mobility

Die Relevanz dieser Problemstellung zeigt sich konkret am Beispiel der Siemens Mobility GmbH:

- **Aktueller Aufwand:** 1,5-2 Stunden manuelle Datenübertragung pro Fahrstraße im 4-Augen-Prinzip
- **Fehlerquellen:** Zahlendreher, Positionsverwechslungen werden oft erst nach Testfahrten erkannt → kostenintensive Nacharbeiten
- **Fehlende Automatisierung:** Keine automatisierte Datenübertragung vom Gleisplan zur Prüftabelle im System verfügbar

Ziel ist eine signifikante Zeitreduktion durch KI-gestützte Automatisierung, wodurch Ingenieure nur noch KI-Ergebnisse validieren müssen statt komplett manuell zu arbeiten.

## 2.2 Problemstellung

In vielen technischen Disziplinen des Ingenieurwesens werden komplexe Systeme und Anlagen noch immer primär über grafische Pläne dokumentiert und kommuniziert. Ob Hydraulikschaltpläne im Maschinenbau, Rohrleitungs- und Instrumentenfließschemata (R&I-Diagramme) in der Verfahrenstechnik, elektrische Schaltpläne in der Elektrotechnik oder Gleispläne in der Bahntechnik – in all diesen Fachbereichen liegt wertvolles technisches Wissen in Form unstrukturierter Vektorgrafiken oder PDF-Dokumente vor [3]. Diese Pläne enthalten eine Vielzahl

semantisch relevanter Informationen: Komponenten werden durch standardisierte oder kunden-spezifische Symbole repräsentiert, Verbindungen zwischen Elementen sind grafisch dargestellt, und technische Spezifikationen werden als Textannotationen beigefügt.

Für moderne digitale Arbeitsabläufe, etwa für die Erstellung eines Digitalen Zwillings, für automatisierte Konsistenzprüfungen oder für die Integration in Enterprise-Resource-Planning-Systeme (ERP, ein betriebswirtschaftliche Softwaresysteme zur integrierten Planung und Steuerung von Ressourcen, Prozessen und Daten über alle Unternehmensbereiche hinweg), müssen diese visuell kodierten Informationen jedoch in strukturierte, maschinenlesbare Datenformate überführt werden. Dieser Medienbruch zwischen rein visuellen oder unstrukturierten Plandarstellungen und semantischen Datenmodellen stellt eine zentrale Herausforderung dar, da die manuelle Übertragung nicht nur zeitaufwendig und fehleranfällig ist, sondern auch kaum skalierbar bleibt, wenn die Komplexität der Systeme oder die Anzahl der zu verarbeitenden Pläne zunimmt [2].

Erschwerend wirkt in der Praxis die hohe Varianz der Plandarstellungen: Unterschiedliche Normen und Hersteller führen zu abweichenden Symbolbibliotheken und heterogenen Layouts. Die Zuordnung von Texten zu Grafiksymbolen folgt dabei oft impliziten Konventionen, die nicht formalisiert sind. Ein weiteres Defizit traditioneller Workflows ist das Fehlen einer bidirektionalen Verknüpfung zwischen den strukturierten Daten und ihrer grafischen Entsprechung im Quelldokument. So lässt sich beispielsweise von einer Tabellenzeile (z. B. Signal AS102) nicht direkt zur exakten Position im Plan navigieren. Diese mangelnde Rückverfolgbarkeit behindert die Validierung massiv, da Ingenieure gezwungen sind, Einträge händisch zu suchen. Auch das Übertragen von Änderungen in Zielsysteme (Excel, Datenbanken, ERP) erfordert hohen Aufwand und birgt durch die fehlende Synchronisierung ein erhebliches Risiko für Dateninkonsistenzen.

### **Anwendungsfall Bahninfrastrukturplanung**

Im konkreten Kontext der vorliegenden Arbeit manifestiert sich diese allgemeine Problemstellung im Bereich der Bahninfrastrukturplanung. Hier werden Gleispläne häufig als PDF- oder Bilddateien bereitgestellt, die Informationen über Stellwerke, Signale, Weichen, Gleiskoppelstellen und Positionsangaben enthalten. Diese grafischen Pläne müssen regelmäßig in strukturierte Datenformate wie Excel, XML oder Datenbanken überführt werden – etwa für Prüfroutinen, Dokumentation oder die Weiterverarbeitung in Planungssystemen.

Aktuell erfolgt dieser Prozess meist manuell, was arbeitsintensiv und fehlerträchtig ist und mit zunehmender Gleisplankomplexität kaum skalierbar bleibt [2]. Typische Gleispläne enthalten 50-200 verschiedene Objekte, die jeweils mit Positionsangaben, Bezeichnungen und technischen Parametern annotiert sind. Die Fehlerquote bei der manuellen Übertragung ist erheblich: Zahlendreher und Positionsverwechslungen werden oft erst nach kostspieligen Testfahrten erkannt. Zudem fehlt die systematische Rückverfolgbarkeit zwischen extrahierten Daten und ihrer Darstellung im Plan, und Änderungen in neuen Planversionen müssen mühsam manuell identifiziert und nachgepflegt werden.

Die zu lösende Herausforderung besteht darin:

- möglichst viele relevante Informationen automatisiert aus unstrukturierten Plänen zu extrahieren,
- die extrahierten Daten in maschinenlesbare Formate zu überführen,
- erkannte Symbole mit ihren zugehörigen Textinformationen semantisch zu verknüpfen,
- Rückverfolgbarkeit zwischen Plan und extrahierten Daten sowie Änderungsverfolgung zwischen Planversionen zu ermöglichen,
- sowie eine benutzerfreundliche Oberfläche für die Validierung und Nachbearbeitung bereitzustellen.

## 2.3 Zielsetzung der Arbeit

Ziel dieser Masterarbeit ist die Entwicklung eines skalierbaren Prototyps zur automatisierten Erkennung, Interpretation und strukturierten Verarbeitung von Gleisplänen. Im Fokus steht dabei die Extraktion von symbolischen und textlichen Inhalten aus Plänen mittels Deep-Learning-basierter Objekt- und Texterkennung. Der entwickelte Prototyp soll dabei so generisch gestaltet werden, dass er prinzipiell auch auf andere technische Zeichnungen (z.B. im Maschinenbau) übertragbar wäre.

Konkret sollen folgende Teilziele erreicht werden:

### 1. Automatisierte Symbolerkennung

Einsatz eines geeigneten Deep-Learning-Modells, um relevante Symbole (Signale, GKS, GM-Blöcke, Koordinaten usw.) präzise in Gleisplänen zu detektieren, unabhängig von Orientierung und Planvarianz.

### 2. Texterkennung

Durchführung einer Textextraktion innerhalb oder neben den erkannten Symbolbereichen, um wichtige Informationen zu extrahieren, einschließlich Vorverarbeitung bei rotierten oder überlagerten Beschriftungen.

### 3. Strukturierte Datenzuordnung

Abbildung der erkannten Objekte und Texte auf semantische Bedeutungen mittels räumlicher Verknüpfungsalgorithmen und regelbasierter Logik.

### 4. Export in strukturierte Formate

Ausgabe der extrahierten und interpretierten Informationen in maschinenlesbare Zielformate wie Excel für nachgelagerte Planungsprozesse.

### 5. Rückverfolgbarkeit & Änderungsverfolgung

Entwicklung von Mechanismen zur Verknüpfung zwischen Bilddaten (PDF) und extrahier-

ten Datenpunkten sowie zur automatisierten Erkennung von Änderungen in unterschiedlichen Planversionen.

### 6. Benutzeroberfläche

Aufbau einer interaktiven Benutzeroberfläche zur Ergebnisdarstellung mit Visualisierung der erkannten Symbole, der zugehörigen Daten und Änderungen sowie der Nachverfolgbarkeit zwischen Daten und Symbolen.

Durch die Umsetzung dieser Ziele soll ein robuster, praxisnaher Demonstrator entstehen, der mit realen Kundendaten anwendbar ist, reproduzierbare Ergebnisse liefert und Potenzial für eine weiterführende Integration in bestehende Prozesse bietet.

## 2.4 Forschungsfragen

Abgeleitet aus der Problemstellung und den Zielsetzungen beschäftigt sich diese Arbeit mit der folgenden zentralen Forschungsfrage:

**HF: Wie lässt sich der Prozess der Datenextraktion aus heterogenen technischen Zeichnungen durch den Einsatz von Deep Learning und hybriden Verarbeitungsstrategien automatisieren, um eine valide Überführung in strukturierte Datenmodelle zu gewährleisten?**

Diese Hauptfrage wird in der vorliegenden Arbeit exemplarisch am Anwendungsfall der Gleisplatine in der Bahninfrastruktur untersucht. Zur Beantwortung werden folgende Teilstudien (TF) adressiert:

- **TF1:** Inwieweit eignen sich aktuelle einstufige Objektdetektoren zur zuverlässigen Erkennung von kleinteiligen, rotierten Symbolen in technischen Zeichnungen und welche Vorverarbeitungsschritte sind notwendig, um die Präzision bei variierenden Layouts zu maximieren?
- **TF2:** Wie können geometrische Informationen und unstrukturierte Textdaten algorithmisch so verknüpft werden, dass eine korrekte semantische Zuordnung (z. B. Signalbezeichnung zu Signalsymbol) auch bei hoher Objektdichte erfolgt?
- **TF3:** Wie muss eine Systemarchitektur gestaltet sein, um neue Symbolklassen und Datenformate modular zu integrieren, wobei Änderungen auf einzelne Pipeline-Komponenten beschränkt bleiben?
- **TF4:** Welcher algorithmische Ansatz eignet sich, um in rein visuellen Daten semantische Änderungen zwischen zwei Planversionen robust zu identifizieren und visualisierbar zu machen?

## 2.5 Forschungsvorgehen

Zur Beantwortung der formulierten Forschungsfragen wurde ein methodischer Ansatz gewählt, der sich aus vier aufeinander aufbauenden Phasen zusammensetzt.

**Phase 1: Literaturrecherche und Technologieanalyse.** Zunächst wurde eine systematische Literaturrecherche in den Bereichen Objekterkennung in technischen Zeichnungen, optische Zeichenerkennung sowie branchenübergreifende Digitalisierung von Planungsunterlagen durchgeführt. Dabei wurden sowohl aktuelle Fachpublikationen aus Datenbanken wie IEEE Xplore, Springer und Google Scholar als auch die offizielle Dokumentation relevanter Open-Source-Werkzeuge (insbesondere YOLOv8, PaddleOCR, Tesseract und EasyOCR) ausgewertet. Die Ergebnisse dieser Phase bilden die Grundlage für Kapitel 3.

**Phase 2: Anforderungserhebung und Domänenanalyse.** Parallel zur Literaturrecherche wurden Experteninterviews mit Fachspezialisten der Siemens Mobility GmbH sowie Rücksprachen mit wissenschaftlichen Betreuern an der TU Clausthal geführt. Diese dienten der Identifikation praxisrelevanter Anforderungen, der Analyse bestehender manueller Arbeitsabläufe sowie dem Verständnis domänenspezifischer Besonderheiten der Gleisplanverarbeitung. Auf dieser Basis wurden die funktionalen und nicht-funktionalen Anforderungen systematisch abgeleitet (Kapitel 4).

**Phase 3: Iterative Konzeption und Implementierung.** Ausgehend von den erhobenen Anforderungen und den identifizierten technologischen Möglichkeiten wurde eine modulare Systemarchitektur konzipiert (Kapitel 5) und prototypisch umgesetzt (Kapitel 6). Die Entwicklung erfolgte iterativ: Einzelne Komponenten wie Objektdetektion, Texterkennung und Verknüpfungsalgorithmen wurden schrittweise implementiert, getestet und auf Basis der Ergebnisse optimiert.

**Phase 4: Evaluation und Validierung.** Die abschließende Evaluation umfasst sowohl komponentenbezogene Leistungskennzahlen als auch eine End-to-End-Bewertung des Gesamtsystems anhand realer Gleispläne (Kapitel 7). Die Validierung erfolgte durch den systematischen Abgleich der automatisiert extrahierten Daten mit manuell erstellten Referenzlisten der Siemens Mobility GmbH.

## 2.6 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in acht Hauptkapitel:

- **Kapitel 1 (Abstract)** liefert eine kurze Zusammenfassung der Zielsetzung, Methodik und zentralen Ergebnisse der Arbeit.
- **Kapitel 2 (Einleitung)** führt in das Thema ein, erläutert die Motivation, beschreibt die zugrunde liegende Problemstellung sowie das Ziel der Arbeit.
- **Kapitel 3 (Theoretische und technische Grundlagen)** stellt die relevanten Konzepte

aus dem Bereich der grafischen Gleispläne, der Symbolik sowie der objekterkennenden Verfahren mit Deep Learning vor.

- **Kapitel 4 (Anforderungsanalyse)** identifiziert die funktionalen und nicht-funktionalen Anforderungen an das System und beschreibt die Herausforderungen in der praktischen Umsetzung.
- **Kapitel 5 (Konzeption des Prototyps)** erläutert die Architektur und den geplanten Workflow des Systems von der PDF-Konvertierung bis zur strukturierten Ausgabe.
- **Kapitel 6 (Implementierung)** beschreibt die technische Realisierung der Komponenten, inklusive des Trainings von Detektionsmodellen, der Texterkennung, des Mappings sowie der Versionierung und Rückverfolgbarkeit.
- **Kapitel 7 (Evaluation)** bewertet das System anhand verschiedener Testmethoden und diskutiert die Ergebnisse und Validierungen mit Siemens-internen Daten.
- **Kapitel 8 (Diskussion)** fasst die wichtigsten Erkenntnisse zusammen, reflektiert kritisch den Entwicklungsprozess und gibt einen Ausblick auf mögliche Weiterentwicklungen.

### 3. Theoretische und technische Grundlagen

Im Rahmen dieses Kapitels werden die maßgeblichen Grundlagen dargelegt, auf denen das entwickelte System basiert. Dabei umfassen die *theoretischen Grundlagen* die konzeptionellen Aspekte wie die Struktur von Gleisplänen, die Klassifikation bahntechnischer Symbole sowie die Prinzipien der Objekterkennung, während die *technischen Grundlagen* die eingesetzten Technologien und Werkzeuge behandeln. Der Fokus liegt auf drei Kernbereichen, die den in Kapitel 2 definierten Zielen entsprechen: (1) automatisierte Objekterkennung zur Symboldetektion, (2) Textextraktion zur Erfassung von Bezeichnungen und Positionsangaben, sowie (3) strukturierte Informationsverarbeitung zur semantischen Verknüpfung der erkannten Elemente.

#### 3.1 Grafische Gleispläne im Bahnkontext

Die Dokumentation einer Bahnanlage stellt eine komplexe Aufgabe dar, da diese typischerweise aus einer Vielzahl von Gleisabschnitten, Weichen und Signalen besteht. Die manuelle Erfassung aller Anlagendetails erfordert üblicherweise mehrere Wochen Arbeit, was die Skalierbarkeit traditioneller Dokumentationsansätze erheblich einschränkt.[4]

Diese zeitintensive manuelle Arbeit motiviert den Einsatz automatisierter Verfahren zur Datenextraktion. Die folgenden Abschnitte legen die theoretischen Grundlagen dar, auf denen eine solche Automatisierung aufbauen kann.

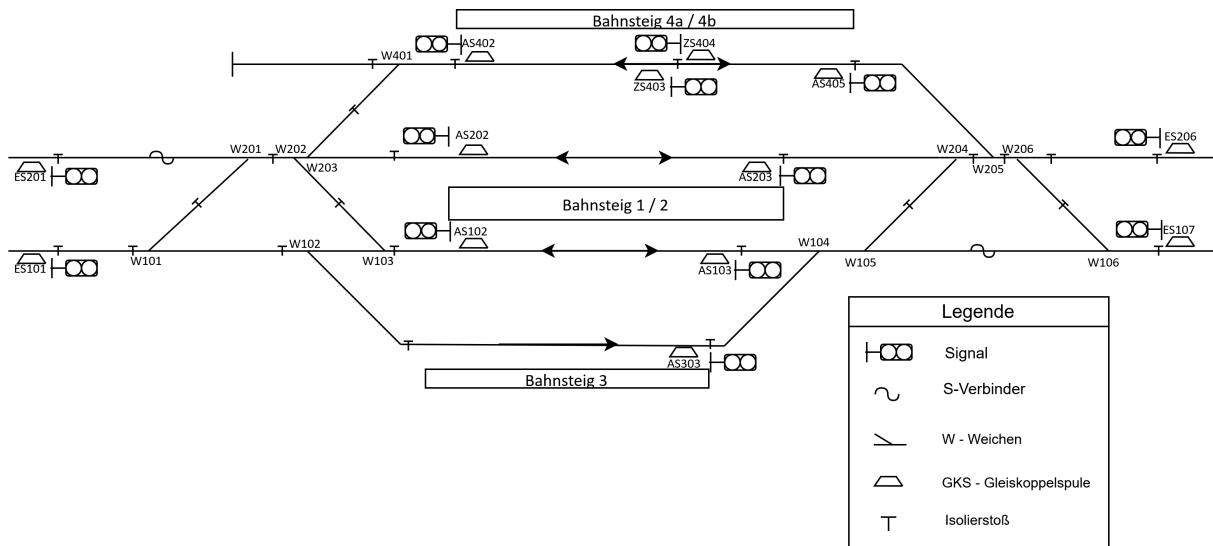
##### 3.1.1 Grundlagen des Gleisplans

Der Gleisplan bildet die fundamentale Datengrundlage für die Planung, den Bau und den operativen Betrieb von Bahnanlagen. Er repräsentiert die technische Infrastruktur und dient als visuelle Schnittstelle zwischen der physischen Außenanlage und der sicherungstechnischen Logik (Stellwerk).

Gleispläne existieren in zwei Darstellungsformen [5]: maßstäbliche Lagepläne (topografisch, bautechnisch) und schematische Übersichtspläne (topologisch, sicherungstechnisch). Diese Arbeit fokussiert letztere, da sie die logischen Fahrwegbeziehungen priorisieren und die Grundlage für die Stellwerksplanung bilden.

Gleispläne enthalten drei Symbolkategorien [5]: (1) Fahrwegelemente (Gleise, Weichen), (2) Sicherungstechnik (Signale, Achszähler), (3) Metadaten (Bezeichner, Kilometrierung). Jedes Element trägt eindeutige Labels zur Identifikation.

Obwohl diese Pläne in der heutigen Ingenieurspraxis mittels CAD-Werkzeugen (Computer Aided Design) erstellt werden, liegt die primäre Herausforderung für eine automatisierte Auswertung nicht im Dateiformat, sondern in der korrekten semantischen Interpretation dieser abstrahierten Symbolik und ihrer logischen Verknüpfung [5].



**Abbildung 3.1:** Ausschnitt eines schematischen Gleisplans mit drei Bahnsteigbereichen (4a/4b, 1/2, 3), wobei Richtungspfeile die zulässigen Fahrtrichtungen kennzeichnen (Beispielhafte Darstellung)

Abbildung 3.1 zeigt einen typischen Bahnhofsgebiet mit drei Bahnsteigen (4a/4b, 1/2, 3) und der zugehörigen Infrastruktur. Die Darstellung umfasst verschiedene bahntechnische Symbole.

**Signale** werden durch ein einheitliches Grundsymbol dargestellt und durch ihre Bezeichnung in drei Kategorien unterteilt: **Einfahrtsignale (ES)** regeln die Einfahrt in den Bahnhof, **Ausfahrtsignale (AS)** regeln die Ausfahrt, und **Zwischensignale (ZS)** regeln Fahrten innerhalb des Bahnhofsgebietes. Die **Weichen (W)** ermöglichen die flexible Fahrweggestaltung zwischen den verschiedenen Gleisen. **Gleiskoppelpulen (GKS)**, als trapezförmige Symbole dargestellt, dienen der punktförmigen Zugbeeinflussung. Die elektrische Trennung von Gleisabschnitten zur Belegungserkennung wird durch **Isolierstöße** (T-förmige Symbole) realisiert. Ergänzend sind **S-Verbinder** (S-förmige Symbole) dargestellt, die Schienenverbindungen zwischen Gleisabschnitten markieren. **Richtungspfeile** kennzeichnen die zulässigen Befahrrichtungen der einzelnen Gleise – beispielsweise zeigt die bidirektionale Pfeildarstellung bei Bahnsteig 1/2 die Befahrbarkeit aus beiden Richtungen an [5].

### 3.1.2 Bahntechnische Symbole und ihre Klassifikation

In den Gleisplänen kommen zahlreiche Symbole zum Einsatz, um die verschiedenen technischen Objekte, Anlagenkomponenten und zusätzlichen Informationen übersichtlich darzustellen [5]. Gleispläne dienen als fundamentale Arbeitsgrundlage für das Engineering von Stellwerken und bilden die zentrale Planungsbasis für verschiedenste Gewerke der Bahntechnik [5, 6]. Stellwerke nutzen diese Pläne beispielsweise für das Einstellen von Signalen mit Fahrstraßenelementen und Ausschlüssen, während die Zugsicherung auf Basis dieser Pläne die Betrachtung der Durchrutschwege und Fahrwegelemente durchführt [5]. Die Zugsicherungssysteme werden dabei vom Stellwerk gesteuert, entweder direkt über die Stellwerkslogik oder indirekt über nachgelagerte Systeme [5]. Alle dargestellten Elemente wie Signale, Weichen,

Isolierstöße, Achszähler, Gleiskoppelpulen, Gleismagnete und S-Verbinder sind Bestandteile der bahntechnischen Infrastruktur, wobei jedes Gewerk abhängig von seiner spezifischen Aufgabenstellung unterschiedliche Teilmengen dieser Elemente benötigt [5]. Da der Gleisplan somit eine zentrale Schnittstelle zwischen technischer Planung, betrieblicher Umsetzung und sicherheitstechnischen Systemen darstellt, ist eine präzise, klare und einheitliche Symbolklassifizierung notwendig [7]. Die für diese Arbeit relevanten Symbolklassen gliedern sich in vier funktionale Kategorien (siehe Tabelle 3.1):

Kategorie	Elemente und Funktion
Stellwerkselemente	<b>Signale:</b> Fahrerlaubnis und Geschwindigkeitsvorgabe[5]. <b>Weichen:</b> Fahrwegverzweigungen[5].
Zugbeeinflussung	<b>Balisen, Gleismagnete (GM), Gleiskoppelpulen (GKS):</b> Übertragung von Informationen (Fahrt-/Haltinformationen, ggf. auch Geschwindigkeitsvorgaben und weitere Daten) vom Stellwerk an den Zug[5].
Belegungserkennung	<b>Isolierstöße:</b> Abgrenzung von Gleisabschnitten zur Belegungserkennung mittels Gleisstromkreisen[5]. <b>Achszähler:</b> Zählpunktbasierte Belegungserkennung durch Erfassung ein- und ausfahrender Achsen[5].
Metadaten	<b>Positionsangaben:</b> Kilometrierung, Lagepunkte[6]. <b>Gleisabschnittsnamen:</b> Eindeutige Gleisidentifikation[6].

**Tabelle 3.1:** Klassifikation bahntechnischer Symbole nach funktionalen Kategorien

Die automatisierte Analyse visueller Informationen bildet das Fundament zahlreicher technischer Anwendungen von der Qualitätskontrolle in der industriellen Fertigung über die medizinische Bildanalyse bis hin zur autonomen Navigation. Während Menschen mühelos in der Lage sind, Objekte in Bildern zu identifizieren und zu lokalisieren, erfordert die Replikation dieser Fähigkeit in Maschinen die Lösung komplexer Teilprobleme. Historisch wurde diese Aufgabe durch explizite Programmierung von Erkennungsregeln adressiert. Der konzeptionelle Durchbruch gelang durch maschinelles Lernen, insbesondere durch tiefe neuronale Netze, die relevante Merkmale selbstständig aus Trainingsdaten extrahieren können [8, 9]. Diese Methodik bildet das Fundament für die in dieser Arbeit entwickelte Symbol- und Texterkennung in Gleisplänen.

Für technische Anwendungen wie die Analyse von Konstruktionszeichnungen oder Gleisplänen manifestieren sich spezifische Herausforderungen, die über die Verarbeitung natürlicher Fotografien hinausgehen. Im Folgenden werden zunächst die besonderen Eigenschaften technischer CAD-Zeichnungen wie hohe Objektdichte, standardisierte Symbolik und beliebige Symbolorientierung erläutert, bevor die grundlegenden methodischen Ansätze der Objekterkennung systematisch hergeleitet werden. Das Kapitel folgt dabei einer Progression vom Allgemeinen zum Spezifischen: Ausgehend von universellen Erkennungsprinzipien wird schrittweise die Anwendung auf bahntechnische Gleispläne konkretisiert.

### 3.1.3 Objekterkennung in technischen CAD-Zeichnungen

Im Gegensatz zu natürlichen Szenen weisen technische Konstruktionszeichnungen spezifische Charakteristika auf, die dedizierte Lösungsansätze erfordern. Diese domänenpezifischen Besonderheiten haben direkten Einfluss auf die Wahl geeigneter Erkennungsverfahren und definieren die in Kapitel 4 formulierten funktionalen Anforderungen an das System.

#### Domänenpezifische Eigenschaften

Technische Zeichnungen unterscheiden sich fundamental von natürlichen Bildern in ihrer Entstehung und Struktur [10]. CAD-Systeme erzeugen vektorbasierte Repräsentationen, die geometrische Primitive wie Linien, Kreise und Polylinien sowie zugeordnete Metadaten enthalten. Die Konvertierung in Rasterformate (PDF, PNG) für ML-basierte Analyse führt jedoch zu einem Informationsverlust hinsichtlich topologischer Beziehungen und semantischer Layer-Informationen. Topologische Beziehungen beschreiben die Verbindungsstruktur zwischen Elementen – etwa welche Gleisabschnitte durch eine Weiche verbunden sind oder welches Signal zu welchem Gleis gehört. Semantische Layer-Informationen kodieren die funktionale Zuordnung von Objekten zu Kategorien wie Gleisführung, Signaltechnik oder Oberleitung. Bei der Rasterisierung werden diese Strukturinformationen eliminiert; das Bild wird zu einer Ansammlung von Pixeln ohne inhärente Bedeutung. Diese Transformation von strukturierten zu unstrukturierten Daten motiviert den Einsatz von Deep-Learning-Verfahren, die robuste Mustererkennung auch ohne explizite topologische Information ermöglichen.

Bahntechnische Symbole folgen standardisierten Bibliotheken gemäß EN-Normen [11] oder kundenspezifischen Richtlinien. Die Variabilität innerhalb einer Symbolklasse ist somit deutlich geringer als bei natürlichen Objekten wie Fahrzeugen oder Personen, jedoch erschweren kundenspezifische Anpassungen die Generalisierung über verschiedene Planungsunternehmen hinweg. Ein durchschnittlicher Bahnhofsgleisplan enthält zwischen 100 und 300 relevante Symbole (Signale, Zugbeeinflussungselemente, Koordinaten) auf einer Fläche von wenigen Quadratmetern, was eine deutlich höhere Objektdichte als in typischen Computer-Vision-Benchmark-Datensätzen wie COCO [12] oder Pascal VOC [13] darstellt. Zum Vergleich: COCO-Bilder enthalten im Durchschnitt 7 bis 10 Objekte, während ein Gleisplanausschnitt von vergleichbarer Pixelgröße 50 bis 100 Symbole aufweisen kann.

Anders als bei Objekten mit kanonischer Orientierung – etwa Straßenverkehrsschilder (typischerweise frontal) oder Fußgänger (vertikal stehend) – können bahntechnische Symbole und Textannotationen entlang der Gleisachsen in beliebigen Winkeln orientiert sein. Ein WeichenSymbol kann beispielsweise in 16 verschiedenen Rotationen auftreten, die jeweils  $22,5^\circ$  Schritte zwischen  $0^\circ$  und  $360^\circ$  abdecken. Signale werden parallel zur Gleisachse positioniert, Gleisnummern folgen der Krümmung des Gleises. Diese Rotationsvariabilität ohne bevorzugte Ausrichtung erfordert rotationsinvariante Detektionsarchitekturen, deren Notwendigkeit in Abschnitt 3.2 detailliert erläutert wird.

## Stand der Forschung

Die Anwendung von Machine Learning auf technische Zeichnungen ist ein aktives Forschungsfeld mit Anwendungen in verschiedenen Ingenieurdisziplinen. Ahmed et al. [14] entwickelten frühe Ansätze zur automatischen Segmentierung von architektonischen Grundrissen unter Verwendung morphologischer Operationen und Hough-Transformation. Moreno-García et al. [15] entwickelten ein umfassendes Framework zur Digitalisierung komplexer technischer Zeichnungen und demonstrierten die Überlegenheit von Deep-Learning-Ansätzen gegenüber klassischen Feature-Engineering-Methoden. Rahul et al. [16] extrahierten Informationen aus P&ID-Diagrammen der Prozessindustrie mittels Faster R-CNN und erreichten dabei mean Average Precision-Werte von über 85%.

Jamieson et al. [17] liefern eine umfassende Übersicht über Deep-Learning-Methoden für Engineering-Diagramme und identifizieren als zentrale Herausforderungen die hohe Variabilität kundenspezifischer Symbolbibliotheken, den Mangel an annotierten Trainingsdaten sowie die Notwendigkeit rotationsinvarianter Detektionsverfahren. Für den bahntechnischen Bereich existieren bisher hauptsächlich proprietäre Lösungen einzelner Eisenbahnunternehmen ohne publizierte wissenschaftliche Evaluierung. Die vorliegende Arbeit adressiert diese Forschungslücke durch Entwicklung und Evaluation eines prototypischen Systems auf Basis moderner One-Stage-Detektoren (Detektoren wie YOLO, die Objekte in einem einzigen Netzwerkdurchlauf lokalisieren und klassifizieren).

### 3.1.4 Entwicklung der Objekterkennung

Die erwähnten Detektionsverfahren basieren auf *Deep Learning*, einer Form des maschinellen Lernens, bei der künstliche neuronale Netze – insbesondere *Convolutional Neural Networks* (CNNs) – relevante Merkmale automatisch aus Trainingsbildern lernen [18]. Im Gegensatz zu klassischen Verfahren, bei denen Erkennungsmerkmale manuell definiert werden müssen, ermöglichen CNNs die End-to-End-Optimierung des gesamten Erkennungssystems. Die folgenden Abschnitte beschreiben zunächst die historische Entwicklung dieser Technologie, bevor spezialisierte Architekturen für die Objekterkennung detailliert werden. Die Objekterkennung hat sich von regelbasierten Verfahren über klassisches maschinelles Lernen zu Deep-Learning-Ansätzen entwickelt [19]. Für die vorliegende Arbeit werden ausschließlich CNN-basierte Detektoren eingesetzt, da diese bei technischen Zeichnungen signifikant höhere Erkennungsraten erreichen als klassische Verfahren (vgl. Kapitel 5, Tabelle 5.2).

### 3.1.5 Architekturen für Deep-Learning-basierte Objekterkennung

Die Implementierung von CNN-basierten Objektdetektoren lässt sich in zwei Architekturparadigmen unterteilen, die sich in ihrer Balance zwischen Detektionsgenauigkeit und Recheneffizienz unterscheiden [19]. Die Wahl der Architektur hat direkte Auswirkungen auf die Praxistauglichkeit des Systems.

### Zweistufige vs. einstufige Detektoren

Zweistufige Detektoren wie Faster R-CNN [20] separieren den Erkennungsprozess konzeptionell in zwei aufeinanderfolgende Phasen. Zunächst generiert ein Region Proposal Network (RPN) zwischen 100 und 300 Kandidatenregionen, die potenziell Objekte enthalten könnten. Diese Regionen werden anschließend durch ein separates Klassifikationsnetzwerk evaluiert, das sowohl die Objektklasse bestimmt als auch die Bounding-Box-Koordinaten verfeinert. Der Vorteil liegt in hoher Detektionspräzision, insbesondere bei kleinen oder teilweise verdeckten Objekten. Die sequenzielle Verarbeitung hunderter Kandidatenregionen führt jedoch zu Bildraten von lediglich 5 bis 10 Bildern pro Sekunde (Frames Per Second, FPS) auf moderner GPU-Hardware (Graphics Processing Unit).

Für die Gleisplananalyse erweist sich diese Geschwindigkeit als kritisch. Ein durchschnittlicher Plan umfasst nach der Kachelungsstrategie (siehe Abschnitt 6.2.3) zwischen 150 und 300 Bildausschnitte. Während YOLO-Modelle typischerweise mit Eingangsgrößen von  $640 \times 640$  oder  $1024 \times 1024$  Pixeln arbeiten, werden für die Gleisplananalyse aufgrund der sehr kleinen Symbolgrößen Kacheln von  $2048 \times 2048$  Pixeln verwendet (Details zur Kachelungsstrategie in Abschnitt 6.2.3). Bei einer Verarbeitungszeit von 100 bis 200 Millisekunden pro Kachel würde die Gesamtanalyse zwischen 4 bis 15 Minuten dauern. Bei größeren Eingabebildern erfolgt ein automatisches Downsampling auf die konfigurierte Eingabegröße, wobei die Ausgabekoordinaten entsprechend zurückskaliert werden.

Einstufige Detektoren wie YOLO (You Only Look Once) [21] adressieren diese Geschwindigkeitsproblematik durch eine fundamental andere Strategie. YOLO unterteilt das Eingabebild in ein regelmäßiges Raster von Zellen (beispielsweise  $13 \times 13$  oder  $26 \times 26$  Zellen bei kleineren Eingabegrößen, bzw. entsprechend größere Raster bei höheren Auflösungen). Für jede Zelle sagt das Netz in einem einzigen Vorwärtsdurchlauf direkt mehrere Aspekte vorher: Liegt in dieser Zelle ein Objekt? Wenn ja, welcher Objektklasse gehört es an? Wie hoch ist die Konfidenz dieser Vorhersage? Und wo genau innerhalb der Zelle befindet sich das Objekt? Diese parallele Verarbeitung aller Bildbereiche ermöglicht Inferenzgeschwindigkeiten von 40 bis über 100 Bildern pro Sekunde [22]. Moderne Varianten wie YOLOv8 erreichen durch architektonische Verbesserungen – insbesondere Feature Pyramid Networks (FPN) [23] für Multi-Scale-Detektion und verbesserte Loss-Funktionen – vergleichbare oder sogar überlegene Präzision gegenüber zweistufigen Detektoren bei Beibehaltung der Echtzeitfähigkeit. FPN ermöglicht dabei die gleichzeitige Erkennung von Symbolen unterschiedlicher Größe, was für Gleispläne essentiell ist, da kleine Elemente wie Isolierstöße und große Symbole wie Signale im selben Bild detektiert werden müssen.

Die Abwägung zwischen beiden Paradigmen muss im Kontext der Anwendungsanforderungen erfolgen. Für technische Gleispläne mit gut sichtbaren, scharf gerenderten Symbolen rechtfertigt die hohe Verarbeitungsgeschwindigkeit marginale Genauigkeitseinbußen. Einstufige Architekturen können bei technischen Zeichnungen vergleichbare Präzision erreichen.

Abbildung 3.2 illustriert den fundamentalen Unterschied zwischen beiden Architekturparadigmen.

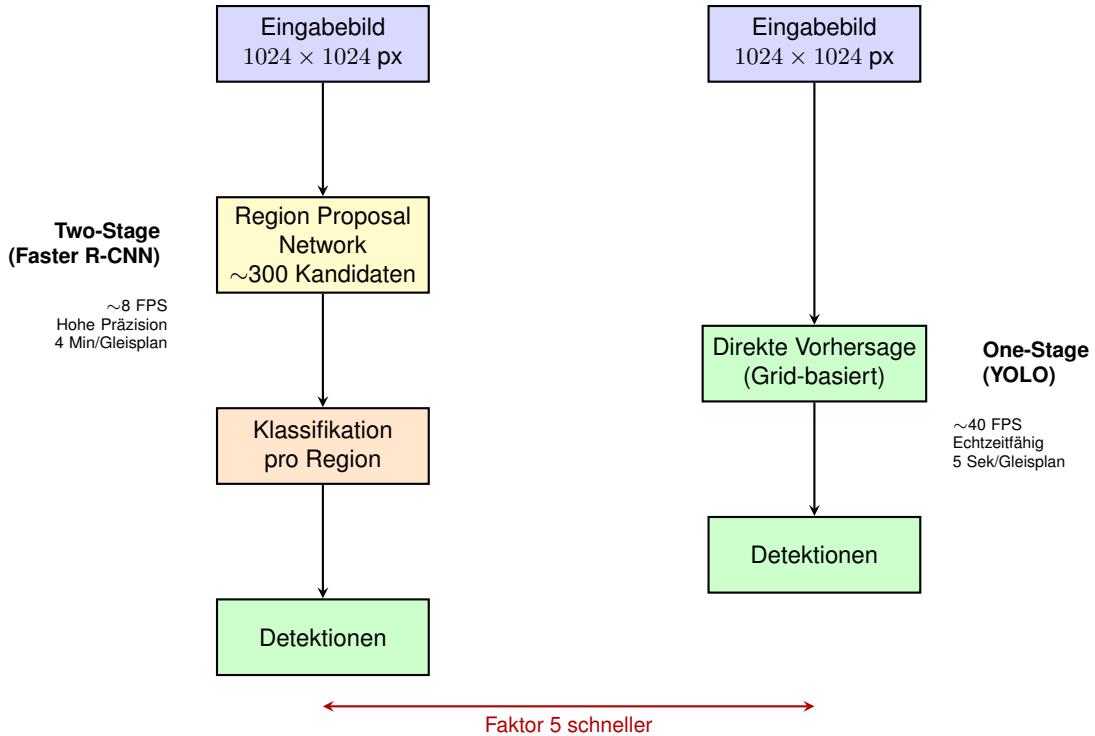


Abbildung 3.2: Architekturvergleich zwischen zweistufigen und einstufigen Objektdetektoren.

### 3.1.6 Modelltraining

Das Training moderner Objektdetektoren wie YOLOv8 erfolgt überwacht (supervised) mit annotierten Bounding Boxes [22]. Jedes Trainingsbeispiel besteht aus einem Eingabebild und den zugehörigen Ground-Truth-Annotationen, die Position, Größe und Klassenzugehörigkeit jedes Objekts spezifizieren.

**Transfer Learning.** Anstatt ein Modell von Grund auf zu trainieren, wird typischerweise mit vortrainierten Gewichten initialisiert [24]. YOLOv8-Modelle werden standardmäßig auf dem COCO-Datensatz (Common Objects in Context) mit 80 Objektklassen vortrainiert. Dieses Transfer Learning ermöglicht es, bereits erlernte low-level Features (Kanten, Texturen) auf die Zieldomäne zu übertragen und beschleunigt die Konvergenz erheblich, insbesondere bei kleineren domänenspezifischen Datensätzen wie sie für Gleispläne typisch sind.

**Datenaugmentation.** Zur Verbesserung der Generalisierungsfähigkeit und Vermeidung von Overfitting setzt YOLOv8 verschiedene Augmentationstechniken ein [25]:

- *Mosaic*: Kombiniert vier Trainingsbilder zu einem, wodurch das Modell lernt, Objekte in verschiedenen Kontexten zu erkennen
- *MixUp*: Überlagert zwei Bilder mit ihren Labels, was die Entscheidungsgrenzen glättet
- *Geometrische Transformationen*: Rotation, Skalierung und Spiegelung erhöhen die Varianz der Trainingsdaten

- *Farbaugmentation*: Variation von Helligkeit, Kontrast und Sättigung verbessert die Robustheit gegenüber unterschiedlichen Scanqualitäten

Die kombinierte Loss-Funktion (vgl. Abschnitt 3.2.4) optimiert simultan Lokalisierung und Klassifikation. Die konkreten Trainingsparameter und domänspezifischen Anpassungen für die Gleisplanerkennung werden in Kapitel 6, Abschnitt 6.2.2 dokumentiert.

### 3.1.7 Evaluationsmetriken

Die Bewertung von Objektdetektoren erfordert standardisierte Metriken, die sowohl Lokalisierungsgenauigkeit als auch Klassifikationsgüte quantifizieren. Im Gegensatz zu einfachen Klassifikationsproblemen müssen Detektoren beide Aspekte simultan korrekt vorhersagen. Die nachfolgenden Metriken bilden die Grundlage für die Systemevaluation.

#### Intersection over Union

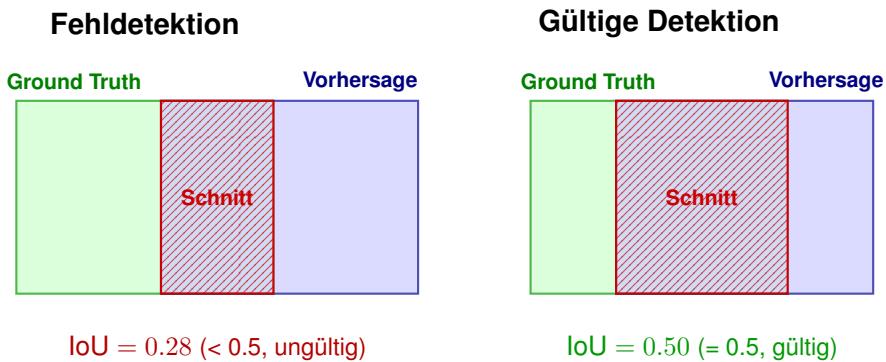
Die Intersection over Union (IoU) [13] quantifiziert den Überlappungsgrad zwischen vorhergesagter Box  $B_{\text{pred}}$  und Ground-Truth-Box  $B_{\text{gt}}$ :

$$\text{IoU}(B_{\text{pred}}, B_{\text{gt}}) = \frac{\text{Area}(B_{\text{pred}} \cap B_{\text{gt}})}{\text{Area}(B_{\text{pred}} \cup B_{\text{gt}})} \quad (3.1)$$

Der IoU-Wert liegt im Intervall  $[0, 1]$ , wobei 0 keine Überlappung und 1 eine perfekte Übereinstimmung signalisiert. In der Praxis gilt eine Detektion als korrekt, wenn  $\text{IoU} \geq 0.5$  – das heißt, die Überlappung muss mindestens die Hälfte der Gesamtfläche betragen. Dieser Schwellenwert reflektiert einen Kompromiss zwischen Forderung nach Präzision und Toleranz gegenüber geringfügigen Positionsabweichungen. Für präzisionskritische Anwendungen werden teilweise höhere Schwellen (0.75 oder 0.9) verwendet [12].

Abbildung 3.3 veranschaulicht diese Berechnung anhand zweier kontrastierender Beispiele. In beiden Fällen repräsentiert die grüne Box die Ground-Truth-Position eines Symbols, während die blaue Box die Vorhersage des Detektors darstellt. Die Schnittfläche (rot schraffiert) entspricht der Überlappung beider Boxen.

Das linke Beispiel zeigt eine Detektion mit  $\text{IoU} = 0.28$ : Die Vorhersage weist zwar auf das korrekte Objekt hin, verfehlt aber die Ground Truth deutlich. Mit einem IoU-Wert unterhalb der Standardschwelle von 0.5 wird diese Detektion als False Positive gewertet. Das rechte Beispiel demonstriert den Grenzfall mit  $\text{IoU} = 0.50$ : Hier überlappt die Schnittfläche exakt die Hälfte der Vereinigungsfläche, was der minimalen Anforderung für eine korrekte Detektion entspricht. Der visuelle Vergleich verdeutlicht die Strenge der IoU-Metrik: Selbst wenn der Detektor das richtige Objekt identifiziert, führt eine ungenaue Lokalisierung zur Klassifikation als Fehldetektion.



**Abbildung 3.3:** Visualisierung der Intersection over Union (IoU) bei unterschiedlichen Überlappungsgraden.

### Precision, Recall und Mean Average Precision

Basierend auf der IoU-Schwelle werden Detektionen klassifiziert: True Positives (TP, korrekte Detektionen mit  $\text{IoU} \geq 0.5$ ), False Positives (FP, Falschdetektionen oder korrekte Klasse aber  $\text{IoU} < 0.5$ ) und False Negatives (FN, übersehene Objekte). Daraus ergeben sich Precision und Recall [13]:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \quad (3.2)$$

Die Precision gibt an, welcher Anteil der vom Detektor gemeldeten Objekte tatsächlich korrekt ist. Ein hoher Precision-Wert bedeutet wenige Fehlalarme. Der Recall misst, welcher Anteil der im Bild tatsächlich vorhandenen Objekte vom Detektor gefunden wurde. Ein hoher Recall-Wert bedeutet, dass wenige Objekte übersehen werden. Beide Metriken stehen typischerweise in einem Trade-off: Durch Erhöhung der Konfidenzschwelle kann Precision verbessert werden, jedoch auf Kosten des Recalls.

Für Gleisplanerkennung ist insbesondere der Recall kritisch. Ein übersehenes Signal oder eine fehlende Weichenkennzeichnung kann zu gravierenden Fehlern in der nachfolgenden Planung führen – etwa falsche Berechnung von Gleiskapazitäten oder Sicherheitsrisiken. Falsch-detections (niedrige Precision) sind weniger kritisch, da sie in der Validierungsphase (siehe Abschnitt 6.5) durch regelbasierte Filter und Plausibilitätsprüfungen eliminiert werden können. Ein falsch detektiertes Signal an einer unmöglichen Position (etwa mitten auf einem Gleis ohne Haltemöglichkeit) wird durch die semantische Validierung verworfen.

Die Mean Average Precision (mAP) [13, 12] aggregiert die Erkennungsgüte über alle Objektklassen und verschiedene Konfidenzschwellen. Für jede Klasse wird zunächst die Average Precision (AP) als Fläche unter der Precision-Recall-Kurve berechnet. Diese Kurve entsteht, indem die Konfidenzschwelle variiert wird: Bei niedriger Schwelle werden viele Objekte detektiert (hoher Recall, niedrige Precision), bei hoher Schwelle nur sehr sichere Detektionen (niedriger Recall, hohe Precision). Die mAP ist der Mittelwert über alle Klassen:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (3.3)$$

wobei  $N$  die Anzahl der Objektklassen bezeichnet. Üblich ist die Angabe von mAP@0.5 (IoU-Schelle 50%) sowie mAP@[0.5:0.95] (Mittelwert über IoU-Schwellen von 50% bis 95% in 5%-Schritten). Die letztere Metrik ist deutlich strenger und fordert präzisere Lokalisierung.

Die mAP-Metrik ist keine Anforderung, sondern die *Standardevaluationsmetrik* für Objektdetektoren wie YOLO [22]. Sie wird sowohl während des Trainings zur Überwachung des Lernfortschritts als auch zur finalen Bewertung des trainierten Modells verwendet. In der vorliegenden Arbeit dient mAP@0.5 als primäre Evaluationsmetrik für die Objekterkennung (Kapitel 7), da für die Gleisplananalyse eine IoU-Schelle von 50% ausreichend ist: Leichte Ungenauigkeiten in der Boxlokalisierung werden durch nachgelagerte Validierung kompensiert. Ein Signal, das mit  $45^\circ$  statt exakt  $47^\circ$  Rotation detektiert wird, ist für die Planungsaufgabe dennoch korrekt identifiziert.

Die in diesem Abschnitt dargestellten Verfahren, von zweistufigen Detektoren wie Faster R-CNN bis zu einstufigen Architekturen wie YOLO, verwenden achsenparallele Begrenzungsrahmen (Axis-Aligned Bounding Boxes, AABB). Diese Boxen sind durch vier Parameter definiert und ihre Kanten verlaufen stets horizontal und vertikal zum Bildrand. Für viele Anwendungen wie Fußgängererkennung oder Fahrzeugdetektion, bei denen Objekte typischerweise aufrechte oder horizontale Orientierung aufweisen, ist dies ausreichend. Technische Zeichnungen mit beliebig rotierten Symbolen und Textannotationen stellen jedoch besondere Anforderungen, die achsenparallele Boxen nur unzureichend erfüllen. Abschnitt 3.2 führt daher orientierte Begrenzungsrahmen (Oriented Bounding Boxes, OBB) ein, die durch einen zusätzlichen Rotationsparameter präzisere Lokalisierung rotierter Objekte ermöglichen.

## 3.2 Oriented Object Detection für rotierte Objekte

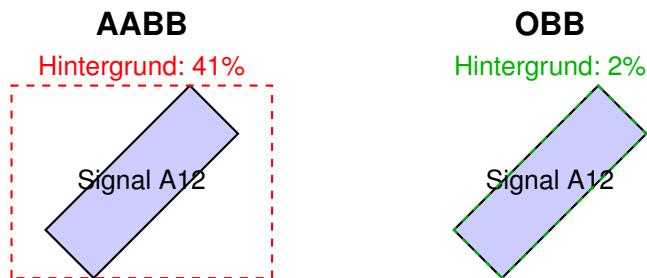
Die in Abschnitt 3.1.5 vorgestellten objekterkennenden Modelle basieren überwiegend auf achsenparallelen Bounding Boxes (Axis-Aligned Bounding Boxes, AABB)[21]. Bei der Detektion rotierter Objekte, wie sie in technischen Zeichnungen und Gleisplänen vorliegen, stoßen AABB-basierte Ansätze jedoch an fundamentale Grenzen. Oriented Bounding Boxes (OBB) bieten hier eine geometrisch präzisere Alternative, indem sie zusätzlich zum Zentrum und den Abmessungen auch den Rotationswinkel des Objekts modellieren.

### 3.2.1 Limitierungen achsenparalleler Bounding Boxes

Achsenparallele Bounding Boxes werden durch vier Parameter definiert: Zentrumskoordinaten ( $c_x, c_y$ ) sowie Breite  $w$  und Höhe  $h$ . Die Orientierung dieser Boxen ist fest an die Bildachsen gekoppelt, was bei rotierten Objekten zu erheblichen Nachteilen führt.

### Hintergrundanteil bei Rotation

Der fundamentale Nachteil von AABB zeigt sich im Verhältnis zwischen der vom Objekt tatsächlich belegten Fläche und der durch die Bounding Box umschlossenen Gesamtfläche. Bei rotierten Objekten umschließen achsenparallele Bounding Boxes einen erheblichen Anteil irrelevanter Hintergrunds[26]. Die AABB muss alle Ecken des rotierten Objekts umschließen und umfasst dabei große Bereiche außerhalb des eigentlichen Objekts. Die OBB hingegen liegt eng am Objekt an und minimiert den Hintergrundanteil. Dieser Unterschied ist in Abbildung 3.4 für ein um  $45^\circ$  rotiertes Textlabel dargestellt.



**Abbildung 3.4:** AABB vs OBB Vergleich bei rotiertem Textlabel.

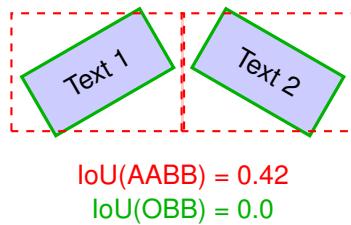
Dieser hohe Hintergrundanteil hat zwei wesentliche Konsequenzen für die Objekterkennung: Erstens verschlechtert sich das Signal-Rausch-Verhältnis der Eingabedaten für nachgelagerte Verarbeitungsschritte wie OCR[27]. Zweitens steigt die Wahrscheinlichkeit falscher Detektionen, da irrelevante Bildelemente innerhalb der Bounding Box liegen[26].

### Non-Maximum Suppression bei dichten Objektgruppen

Ein weiteres Problem ergibt sich bei der Nachbearbeitung von Detektionen mittels Non-Maximum Suppression (NMS)[20]. NMS eliminiert redundante Detektionen durch Unterdrückung überlappender Boxen basierend auf ihrer Intersection over Union (IoU):

$$\text{IoU}(B_1, B_2) = \frac{\text{Area}(B_1 \cap B_2)}{\text{Area}(B_1 \cup B_2)} \quad (3.4)$$

Bei dicht platzierten rotierten Objekten führen AABB zu hohen IoU-Werten, obwohl die Objekte selbst nicht überlappen[26]. Dies resultiert in fälschlicher Unterdrückung korrekter Detektionen. Abbildung 3.5 demonstriert dieses Fehlverhalten anhand zweier nahe beieinanderliegender Textlabels. Die beiden Textobjekte sind um  $30^\circ$  bzw.  $-30^\circ$  rotiert und überlappen sich nicht. Ihre achsenparallelen Bounding Boxes hingegen weisen eine signifikante Überlappung mit  $\text{IoU} = 0.42$  auf. Bei einem typischen NMS-Schwellenwert von  $\tau = 0.4$ [28] würde eine der beiden korrekten Detektionen fälschlicherweise unterdrückt. Im Gegensatz dazu berechnet sich die IoU zwischen den orientierten Bounding Boxes korrekt zu 0.0, da die Objekte tatsächlich nicht überlappen.



**Abbildung 3.5:** NMS-Problem bei dicht platzierten rotierten Objekten.

### 3.2.2 Oriented Bounding Boxes als Lösung

Oriented Bounding Boxes erweitern die AABB-Präsentation um einen zusätzlichen Freiheitsgrad: den Rotationswinkel  $\theta$ . Eine OBB wird durch fünf Parameter definiert:

$$\text{OBB} = (c_x, c_y, w, h, \theta) \quad (3.5)$$

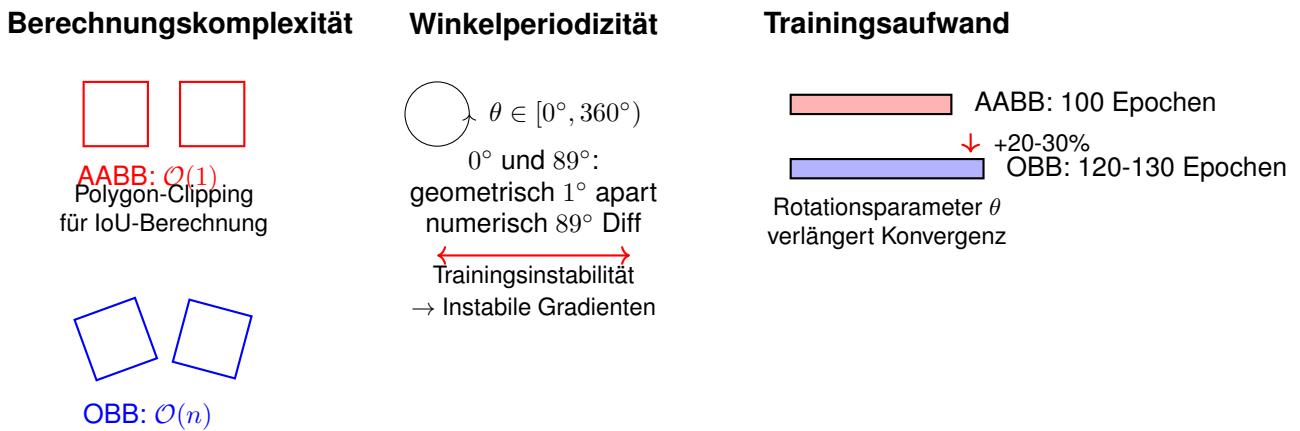
wobei  $(c_x, c_y)$  das Zentrum,  $w$  und  $h$  die Dimensionen entlang der lokalen Achsen und  $\theta \in [-90^\circ, 90^\circ]$  den Rotationswinkel relativ zur horizontalen Bildachse beschreiben. Diese Parametrisierung ermöglicht eine eng anliegende Umhüllung rotierter Objekte und reduziert den Hintergrundanteil auf das geometrische Minimum.

#### Vorteile für rotierte Objekte

Die präzisere geometrische Modellierung durch OBB bietet mehrere Vorteile [29]. Die IoU zwischen OBB reflektiert die tatsächliche Objektüberlappung, wodurch Fehlunterdrückungen bei NMS vermieden werden. Dies ist besonders wichtig bei dicht gruppierten Objekten, wo AABB fälschlicherweise hohe Überlappungen anzeigen würden. Zudem kodiert der Parameter  $\theta$  explizit die Objektorientierung, was für nachgelagerte Prozesse wie orientierungsabhängige OCR wertvoll ist.

### 3.2.3 Herausforderungen bei der Verwendung von OBB

Trotz der geometrischen Vorteile bringen OBB zusätzliche Komplexität in die Detektionspipelinen, die sich auf Training, Inferenz und Nachbearbeitung auswirkt. Abbildung 3.6 fasst die drei Hauptherausforderungen zusammen: erhöhte Berechnungskomplexität bei der IoU-Berechnung, die Periodizität des Rotationswinkels, die zu Trainingsinstabilität führt, und der erhöhte Trainingsaufwand gegenüber AABB-Modellen.



**Abbildung 3.6:** Hauptherausforderungen bei OBB-Verwendung: IoU-Berechnung erfordert Polygon-Clipping statt einfacher Min/Max-Operationen (links), zyklische Winkel führen zu Gradienteninstabilität (Mitte), und der zusätzliche Rotationsparameter erhöht den Trainingsaufwand (rechts).

### Erhöhte Berechnungskomplexität

Die Berechnung der Intersection over Union zwischen zwei OBB ist algorithmisch aufwendiger als bei AABB. Während die AABB-IoU durch einfache Min-Max-Operationen in konstanter Zeit  $\mathcal{O}(1)$  berechnet werden kann, erfordert die OBB-IoU die Lösung eines Polygon-Clipping-Problems[30]:

$$\text{IoU}_{\text{OBB}}(B_1, B_2) = \frac{\text{Area}(P_1 \cap P_2)}{\text{Area}(P_1 \cup P_2)} \quad (3.6)$$

wobei  $P_1$  und  $P_2$  die als Vierecke repräsentierten OBB sind. Die Schnittpunktberechnung zwischen den Kantensegmenten skaliert mit  $\mathcal{O}(n^2)$  für  $n$ -Ecke, was in der Praxis zu einem Faktor 3-5 langsameren NMS-Durchläufen führt[29].

### Winkelperiodizität und Trainingsinstabilität

Der Rotationswinkel  $\theta$  weist eine zyklische Periodizität auf: Eine Rotation um  $180^\circ$  (bzw. bei symmetrischen Objekten um  $90^\circ$ ) resultiert in einer geometrisch identischen Bounding Box[31, 32]. Dies führt zu Ambiguitäten in der Netzwerk-Ausgabe. Betrachtet man beispielsweise zwei Vorhersagen  $\theta_1 = 0^\circ$  und  $\theta_2 = 89^\circ$  für dasselbe Objekt, so sind diese geometrisch nur  $1^\circ$  voneinander entfernt, numerisch jedoch  $89^\circ$ . Standard-Regressionsverluste (z.B. Smooth L1) behandeln diese Diskrepanz linear, was zu instabilen Gradienten führt.

### Erhöhter Trainingsaufwand

Die zusätzliche Parameterregression für  $\theta$  erhöht die Komplexität des Lernproblems. Empirische Studien zeigen, dass OBB-Modelle typischerweise 20-30% mehr Trainingsepochen benötigen als vergleichbare AABB-Modelle, um ähnliche Konvergenzniveaus zu erreichen[29]. Eine

Trainingsepochen bezeichnet dabei einen vollständigen Durchlauf durch den gesamten Trainingsdatensatz, bei dem jedes Trainingsbeispiel einmal zur Gewichtsoptimierung verwendet wird. Dies resultiert aus der Notwendigkeit, sowohl die präzise Lokalisierung als auch die korrekte Orientierungsprädiktion zu erlernen.

### 3.2.4 YOLOv8-OBB Architektur

YOLOv8 bietet eine spezialisierte OBB-Variante, die für die Symbolerkennung geeignet ist. Die Architekturwahl wird in Kapitel 5 begründet. YOLOv8-OBB erweitert die Standard-Architektur um die Regression des Rotationsparameters  $\theta$ , während die grundlegende Netzwerkstruktur aus Backbone, Neck und Detection Heads erhalten bleibt[22]. Die schematische Architektur ist in Abbildung 3.7 dargestellt. Der CSPNet-Backbone extrahiert hierarchische Merkmale und erzeugt eine Feature Pyramid mit drei Auflösungsstufen: P3 ( $128 \times 128$ ) für kleine Objekte wie GKS-Symbole, P4 ( $64 \times 64$ ) für mittlere Objekte und P5 ( $32 \times 32$ ) für große Objekte wie Signale. Das PANet-Neck fusioniert diese Multi-Scale-Features durch bidirektionale Pfade (Top-Down und Bottom-Up), bevor drei spezialisierte Detection Heads die finalen OBB-Vorhersagen ( $c_x, c_y, w, h, \theta, \text{cls}$ ) produzieren.

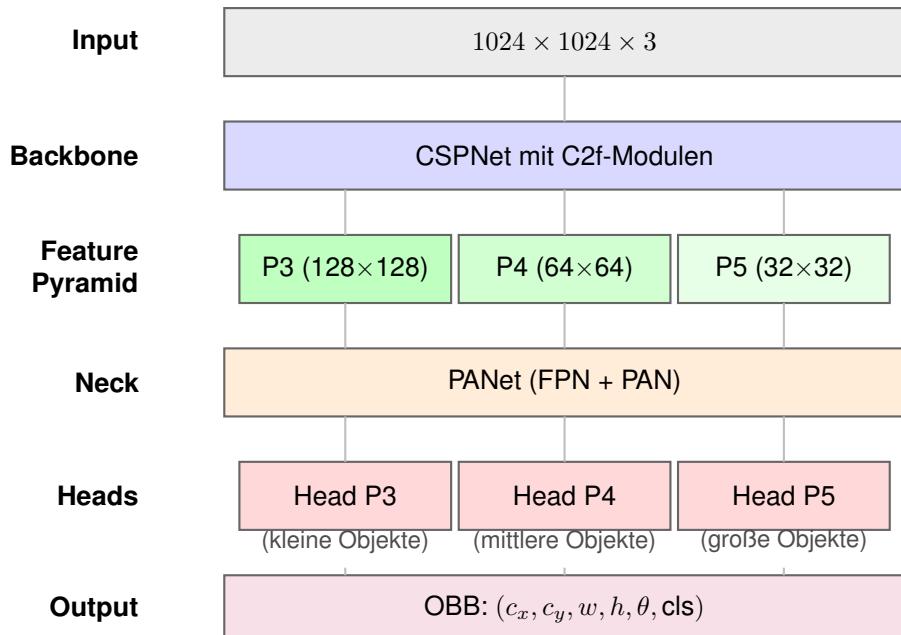


Abbildung 3.7: YOLOv8-OBB Netzwerkarchitektur (nach [22]).

### Netzwerkkomponenten

Die Architektur gliedert sich in drei funktionale Blöcke. Der Backbone extrahiert hierarchische Merkmale aus den Eingabebildern und verwendet eine auf CSPNet basierende Architektur mit Cross Stage Partial Connections[33]. Die sogenannten C2f-Module (CSP Bottleneck with 2 Convolutions, faster) ersetzen die C3-Module früherer Versionen und bieten eine effizientere Gradientenfluss-Charakteristik bei gleichzeitiger Reduktion der Parameteranzahl. Das Neck-

Modul fusioniert Merkmale unterschiedlicher Auflösungsstufen durch eine Kombination aus Top-Down- und Bottom-Up-Pfaden. Die Path Aggregation Network (PANet) Architektur[34] ermöglicht eine effektive Informationspropagation zwischen den Skalen, was für die Detektion sowohl kleiner als auch großer Objekte essentiell ist. YOLOv8 folgt einem Anchor-Free-Ansatz[35], bei dem Objektzentren direkt vorhergesagt werden, anstatt vordefinierte Anker-Boxen zu verwenden. Die Heads sind dezentralisiert (decoupled): Separate Zweige regredieren die Lokalisierung (inklusive  $\theta$ ) und die Klassifikation. Diese Entkopplung verbessert die Konvergenz, da die beiden Aufgaben unterschiedliche Repräsentationen erfordern[36].

### **Verlustfunktion und Training**

Das Training von YOLOv8-OBB optimiert eine kombinierte Verlustfunktion, die drei Komponenten umfasst:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}} \mathcal{L}_{\text{box}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{dfl}} \mathcal{L}_{\text{dfl}} \quad (3.7)$$

Die Box-Loss  $\mathcal{L}_{\text{box}}$  quantifiziert die Lokalisierungsgenauigkeit der vorhergesagten OBB gegenüber der Ground Truth. Die Classification-Loss  $\mathcal{L}_{\text{cls}}$  verwendet Binary Cross-Entropy für die Klassenzuordnung[28]. Focal Loss adressiert dabei das Problem der Klassenungleichverteilung, das bei Gleisplänen auftritt, da manche Symbolklassen (z.B. Koordinaten) deutlich häufiger vorkommen als andere (z.B. Gleismagnete). Die Distribution Focal Loss (DFL)[37] verbessert die Bounding-Box-Regression durch Modellierung der Koordinaten als Verteilungen anstelle von Punktschätzungen.

Die konkreten Trainingsparameter und -ergebnisse für den im Rahmen dieser Arbeit trainierten Detektor werden in Kapitel 6 detailliert dokumentiert. Die Kombination aus Echtzeitfähigkeit, Rotationsinvarianz und Anchor-Free-Design macht YOLOv8-OBB zu einem geeigneten Kandidaten für die Gleisplananalyse. Die endgültige Architekturentscheidung wird in Kapitel 5 dokumentiert.

#### **3.2.5 Relevanz für Gleisplananalyse**

Die beschriebenen OBB-Eigenschaften sind besonders relevant für die automatisierte Analyse von Gleisplänen. Technische Zeichnungen dieses Typs weisen drei charakteristische Merkmale auf, die den Einsatz orientierter Bounding Boxes notwendig machen. Symbole folgen der Topologie der Gleisachsen und können daher in jedem Winkel  $\theta \in [0^\circ, 360^\circ]$  vorliegen. In Bahnhöfen und Weichenstraßen treten Symbole räumlich konzentriert auf, wobei AABB-basierte NMS zu Fehlunterdrückungen führen würde. Die vom OBB-Detektor gelieferte Rotation  $\theta$  ist für nachgelagerte Prozesse wie die orientierungsabhängige OCR (vgl. Kapitel 6.3) essentiell, da Textlabels entsprechend ihrer Ausrichtung verarbeitet werden müssen.

Die in Kapitel 4 definierten funktionalen Anforderungen an die Symbolerkennung erfordern daher zwingend den Einsatz rotationsinvarianter Detektionsverfahren. Die in Kapitel 6 beschriebe-

ne Implementierung nutzt YOLOv8-OBB mit Standard-Konfiguration für die Detektion von 13 domänenspezifischen Symbolklassen (vgl. Tabelle 4.1 in Kapitel 4).

### 3.3 Optical Character Recognition (OCR)

Die optische Zeichenerkennung bildet das verbindende Element zwischen der visuellen Symbolerkennung und der semantischen Interpretation von Gleisplänen. Während die in Abschnitt 3.1.3 beschriebene Objekterkennung Symbole lokalisiert und klassifiziert, liefert sie zunächst keine Information über die zugehörigen Textinhalte. Erst durch die zuverlässige Extraktion von Signalbezeichnungen (z. B. „AS102“), Kilometerangaben (z. B. „18.1606“) und GKS-Kennungen (z. B. „1234“) erhalten die erkannten Symbole ihre vollständige technische Bedeutung.

Dieser Abschnitt behandelt zunächst die allgemeinen Grundlagen der optischen Zeichenerkennung, bevor die spezifischen Herausforderungen technischer Zeichnungen und die für Gleispläne relevanten Lösungsansätze dargelegt werden. Die Progression folgt dabei dem Muster der vorangegangenen Abschnitte: vom allgemeinen Prinzip über moderne Architekturen bis zur domänenspezifischen Anwendung.

#### 3.3.1 Grundlagen und Entwicklung der OCR-Technologie

Die optische Zeichenerkennung (Optical Character Recognition, OCR) bezeichnet die automatisierte Umwandlung von Bildern, die Text enthalten, in maschinenlesbaren Text. Diese Aufgabe lässt sich konzeptionell in vier Verarbeitungsstufen unterteilen: Vorverarbeitung, Textdetektion, Zeichenerkennung und Nachverarbeitung.

Die **Vorverarbeitung** bereitet das Eingabebild für die nachfolgenden Schritte auf. Typische Operationen umfassen Kontrastverbesserung, Rauschunterdrückung und Binarisierung (Umwandlung in Schwarz-Weiß). Bei technischen Zeichnungen ist zusätzlich die Entfernung störender Linienelemente relevant [38].

Die **Textdetektion** lokalisiert Bereiche im Bild, die Text enthalten. Moderne Verfahren verwenden neuronale Netze zur Segmentierung von Textregionen, wobei die Ausgabe typischerweise als Bounding Boxes oder Polygone erfolgt [39].

Die **Zeichenerkennung** wandelt die lokalisierten Bildausschnitte in Zeichenfolgen um. Dies ist die eigentliche „Erkennung“ im engeren Sinne und bildet den rechenintensivsten Schritt der Pipeline.

Die **Nachverarbeitung** validiert und korrigiert die erkannten Texte. Für domänenspezifische Anwendungen wie Gleispläne können hier Musterprüfungen (Regular Expressions) und Plausibilitätskontrollen integriert werden.

### 3.3.2 Architekturen moderner OCR-Systeme

Moderne OCR-Systeme basieren auf zwei dominierenden Architekturparadigmen: der CRNN-Architektur (Convolutional Recurrent Neural Network) und Transformer-basierten Ansätzen. Beide werden im Folgenden erläutert.

#### CRNN-Architektur

Die CRNN-Architektur, eingeführt von Shi et al. [40], kombiniert drei Komponenten zu einem End-to-End-trainierbaren System (Abbildung 3.8). In dieser Arbeit ermöglicht CRNN die Erkennung der alphanumerischen Beschriftungen in Gleisplänen, darunter Signalnamen (z.B. AS102), Koordinatenangaben und GKS-Bezeichnungen.

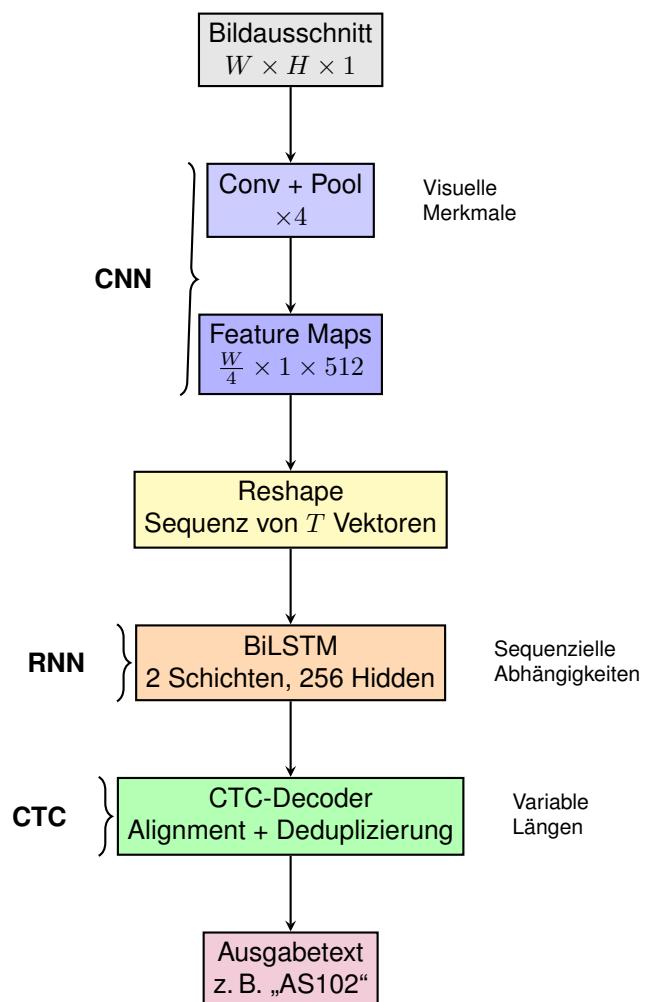


Abbildung 3.8: CRNN-Architektur

**CNN-Komponente (Feature Extraction):** Ein Convolutional Neural Network extrahiert visuelle Merkmale aus dem Eingabebild. Durch sukzessive Faltungs- und Pooling-Operationen wird das Bild auf eine Sequenz von Merkmalsvektoren reduziert. Ein Bildausschnitt der Größe  $W \times H$  wird typischerweise auf  $T = W/4$  Zeitschritte mit je 512 Merkmalskanälen komprimiert[40].

**RNN-Komponente (Sequence Modeling):** Ein bidirektionales LSTM (Long Short-Term Memory) [41] verarbeitet die Merkmalssequenz und modelliert kontextuelle Abhängigkeiten zwischen benachbarten Zeichen. Die Bidirektionalität ermöglicht die Berücksichtigung sowohl vergangener als auch zukünftiger Kontextinformation.

**CTC-Decoder (Transcription):** Die Connectionist Temporal Classification (CTC) [42] löst das Problem variabler Ausgabelängen. Da die Anzahl der CNN-Ausgabespalten nicht mit der Anzahl der Textzeichen übereinstimmt, führt CTC ein „Blank“-Symbol ein und definiert eine Verlustfunktion, die alle möglichen Alignments zwischen Eingabe und Ausgabe marginalisiert:

$$P(y | x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} \prod_{t=1}^T P(\pi_t | x) \quad (3.8)$$

wobei  $y$  die Zielsequenz,  $x$  die Eingabe,  $\mathcal{B}^{-1}(y)$  die Menge aller Pfade bezeichnet, die nach Entfernung von Blanks und Deduplizierung  $y$  ergeben, und  $P(\pi_t | x)$  die Wahrscheinlichkeit des Symbols  $\pi_t$  zum Zeitpunkt  $t$  ist.

Für Maschinenbau-Ingenieure lässt sich die CTC-Funktionsweise anschaulich erklären: Das Netz gibt für jeden Zeitschritt (jede Spalte des Bildes) eine Wahrscheinlichkeitsverteilung über alle möglichen Zeichen aus. Die Ausgabe „A-S-11-0-2“ (wobei „-“ das Blank-Symbol bezeichnet) wird durch Entfernung der Blanks und Zusammenfassung aufeinanderfolgender identischer Zeichen zu „AS102“ dekodiert. Diese Fähigkeit zur Verarbeitung variabler Textlängen ist für Gleispläne essentiell, da Beschriftungen von kurzen Signalnamen (z.B. „AS1“) bis zu längeren Koordinatenangaben (z.B. „18.1606“) reichen.

### Transformer-basierte OCR

Neuere Ansätze ersetzen die RNN-Komponente durch Transformer-Architekturen [43]. Der TrOCR-Ansatz [44] verwendet einen Vision Transformer (ViT) als Encoder und einen Text Transformer als Decoder. Vision Transformer ist ein Transformer Modell, das Bilder als Sequenz von Patches verarbeitet[45]. Der zentrale Mechanismus ist die Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (3.9)$$

wobei  $Q$  (Query),  $K$  (Key) und  $V$  (Value) lineare Projektionen der Eingabe sind und  $d_k$  die Dimensionalität der Keys bezeichnet. Die Division durch  $\sqrt{d_k}$  stabilisiert die Gradienten.

Transformer-basierte Ansätze bieten Vorteile bei langen Sequenzen und ermöglichen effizientere Parallelisierung während des Trainings. Für kurze Texte wie Signalbezeichnungen (typischerweise 3–8 Zeichen) sind die Vorteile jedoch marginal, weshalb CRNN-basierte Systeme wie PaddleOCR in der Praxis häufig bevorzugt werden.

### Vergleich der Architekturen

Tabelle 3.2 fasst die wesentlichen Unterschiede zwischen CRNN und Transformer-basierten Ansätzen zusammen.

Kriterium	CRNN (z. B. PaddleOCR)	Transformer (z. B. TrOCR)
Genauigkeit (kurze Texte)	>95% [46]	>95% [44]
Inferenzzeit (CPU)	8–15 ms/Bild [46]	50–150 ms/Bild <sup>1</sup>
Modellgröße	8–12 MB [46]	300–500 MB [44]
Trainingsdatenbedarf	Moderat (10k–100k Samples)	Hoch (>1M Samples für Pre-training) [44]

**Tabelle 3.2:** Vergleich von CRNN- und Transformer-basierten OCR-Architekturen

### 3.3.3 OCR-Systeme im praktischen Einsatz

Für die praktische Anwendung stehen mehrere etablierte OCR-Frameworks zur Verfügung, die unterschiedliche Stärken aufweisen.

#### Tesseract

Tesseract [47] ist eine Open-Source-Engine, die ursprünglich von Hewlett-Packard entwickelt und später von Google weiterentwickelt wurde. Seit Version 4.0 verwendet Tesseract ein LSTM-basiertes Erkennungsmodell. Die Konfiguration erfolgt über Page Segmentation Modes (PSM), die das erwartete Dokumentlayout spezifizieren:

- **PSM 7:** Einzelne Textzeile – optimal für isolierte Beschriftungen
- **PSM 8:** Einzelnes Wort – für kompakte Labels
- **PSM 13:** Raw Line – für einzelne Textzeilen ohne Layoutsegmentierung, optimal für isolierte Beschriftungen

Eine Whitelist-Funktionalität ermöglicht die Einschränkung des Zeichensatzes auf erwartete Zeichen (z. B. nur Großbuchstaben und Ziffern für Signalbezeichnungen), was die Erkennungs-genaugkeit in domänenspezifischen Anwendungen erhöht.

#### PaddleOCR

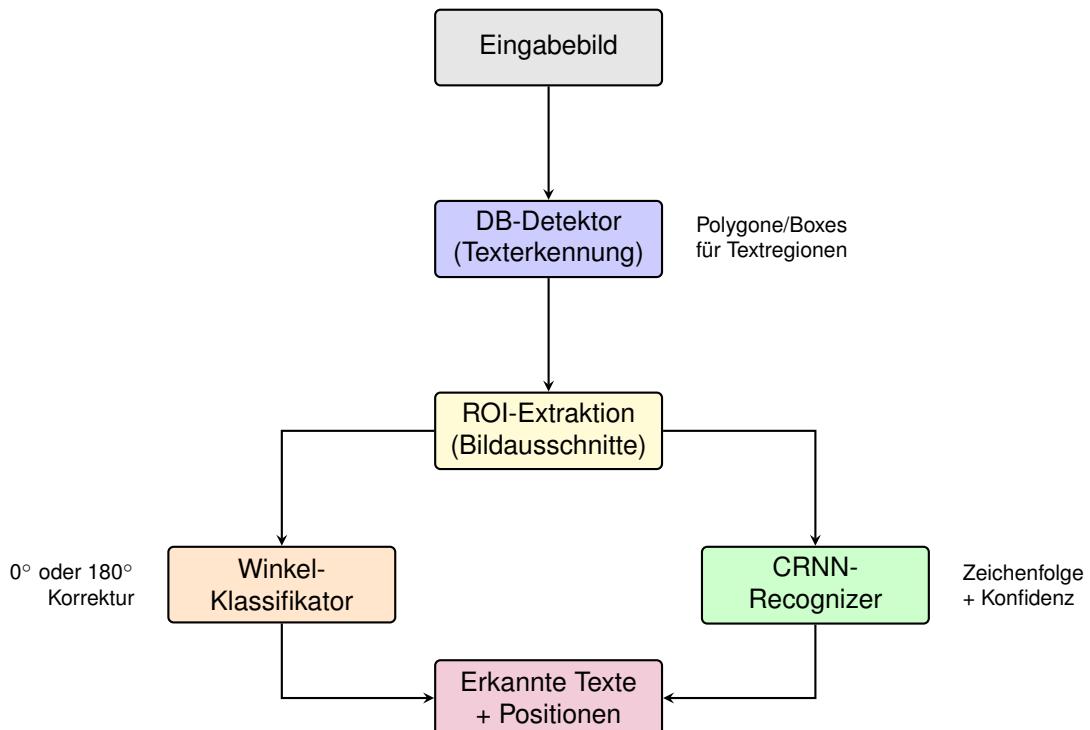
PaddleOCR [46] ist ein vom chinesischen Technologiekonzern Baidu entwickeltes Framework, das eine vollständige Pipeline aus Textdetektion, Winkelklassifikation und Texterkennung bereitstellt. Für die vorliegende Arbeit wurde PaddleOCR gewählt, da es integrierte Winkelerkennung

---

<sup>1</sup>Transformer-basierte Modelle erfordern aufgrund des quadratischen Attention-Mechanismus höhere Inferenzzeiten [43].

für rotierte Texte bietet und effiziente Verarbeitung der zahlreichen Textregionen in Gleisplänen ermöglicht. Die PP-OCRV3-Architektur besteht aus drei Komponenten:

1. **DB-Detektor (Differentiable Binarization)**: Lokalisiert Textregionen durch semantische Segmentierung mit differenzierbarer Schwellenwertberechnung [39].
2. **Winkelklassifikator**: Bestimmt die Orientierung des Textes ( $0^\circ$  oder  $180^\circ$ ) zur automatischen Korrektur.
3. **CRNN-Recognizer**: Erkennt die Zeichenfolge basierend auf der in Abschnitt 3.3.2 beschriebenen Architektur.



**Abbildung 3.9:** PaddleOCR-Pipeline: Dreistufige Architektur mit Detektion, Winkelkorrektur und Erkennung.

### Vergleich der OCR-Systeme

Tabelle 3.3 vergleicht die für technische Zeichnungen relevanten OCR-Systeme.

System	Architektur	Stärken	Schwächen
Tesseract	LSTM-basiert	Whitelist-Funktion, geringer Ressourcenbedarf	Empfindlich bei Rotation, langsam bei vielen ROIs
PaddleOCR	DB + CRNN	Integrierte Winkelerkennung, schnell, robust	Komplexere Installation, größeres Modell
EasyOCR	CRAFT(Character Region Awareness for Text) + CRNN [48]	Einfache API, gute Winkeltoleranz	Langsamer, weniger konfigurierbar
TrOCR	Transformer	State-of-the-Art Genauigkeit	Hoher GPU-Bedarf, langsam

**Tabelle 3.3:** Vergleich etablierter OCR-Systeme für technische Anwendungen

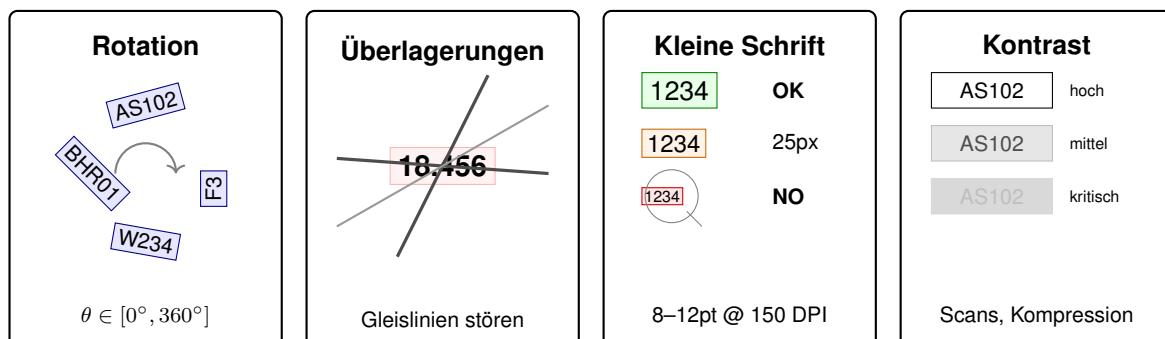
### 3.3.4 Evaluationsmetriken für Texterkennung

Für die isolierte Bewertung von OCR-Systemen existieren etablierte Metriken wie die **Character Error Rate (CER)** und **Word Error Rate (WER)**, die den Anteil fehlerhafter Zeichen bzw. Wörter quantifizieren [49, 50]. Die **Feldgenauigkeit** (Field Accuracy) bewertet, ob ein vollständiges Textfeld exakt korrekt erkannt wurde.

Für integrierte Extraktionspipelines, bei denen OCR nur eine von mehreren Stufen darstellt (Detektion → OCR → Verknüpfung → Validierung), ist jedoch eine *End-to-End-Bewertung* aussagekräftiger: Sie misst, ob das finale Extraktionsergebnis korrekt ist, unabhängig davon, in welcher Pipeline-Stufe Fehler auftraten oder kompensiert wurden. Dieser Ansatz wird in der vorliegenden Arbeit verfolgt.

### 3.3.5 Herausforderungen bei technischen Zeichnungen

Technische Zeichnungen wie Gleispläne stellen OCR-Systeme vor spezifische Herausforderungen, die über typische Dokumentenscans hinausgehen. Diese Herausforderungen werden in Abbildung 3.10 visualisiert.



**Abbildung 3.10:** Zentrale Herausforderungen der OCR in technischen Zeichnungen.

### Variable Textorientierung

In Gleisplänen folgen Beschriftungen der Topologie der Gleise und können in beliebigen Winkeln  $\theta \in [0^\circ, 360^\circ]$  orientiert sein. Standard-OCR-Engines sind primär für horizontalen Text ( $0^\circ$ ) optimiert und zeigen bei geneigten Texten signifikante Leistungseinbußen [51].

Lösungsansätze umfassen:

- **Vorrotation:** Transformation des Bildausschnitts basierend auf dem von der Objekterkennung gelieferten Winkel  $\theta$
- **Multi-Winkel-Inferenz:** Sequenzielle Verarbeitung mit  $0^\circ, 90^\circ, 180^\circ, 270^\circ$  Rotation und Auswahl des konfidentesten Ergebnisses
- **Rotationsinvariante Netze:** Verwendung von Spatial Transformer Networks (STN) [52] zur lernbasierten Entzerrung

### Grafische Überlagerungen

Technische Zeichnungen enthalten zahlreiche Linienelemente (Gleise, Bemaßungen, Führungslien), die Textbereiche durchkreuzen können. Diese Überlagerungen führen zu Segmentierungsfehlern und Fehlerkennungen.

Lösungsansätze umfassen:

- **Morphologische Operationen:** Opening-Filter mit linienförmigen Strukturelementen zur Entfernung dünner Linien [53]. Die Implementierung verwendet orientierte Strukturelemente, die dem Textwinkel folgen, um schräge Linien gezielt zu entfernen ohne Textstriche zu beschädigen.
- **Farbbasierte Filterung:** Separation von Textfarbe (typischerweise Schwarz) und Linienfarben (oft Grau oder Farbe)

Theoretische Alternativen wie Inpainting [54] zur Rekonstruktion überlagerter Bereiche wurden evaluiert, jedoch aufgrund des höheren Rechenaufwands und der ausreichenden Wirksamkeit morphologischer Operationen nicht implementiert.

### Kleine Textgrößen

Beschriftungen in Gleisplänen sind oft nur 8–12 Punkt groß. Bei einer Digitalisierung mit 150 DPI entspricht dies einer Zeichenhöhe von lediglich 17–25 Pixeln. Für zuverlässige OCR werden typischerweise mindestens 30–40 Pixel Zeichenhöhe benötigt [47, 46].

Die Beziehung zwischen Schriftgröße  $s$  (in Punkt), Scanauflösung  $r$  (in DPI) und resultierender Pixelhöhe  $h_c$  ist:

$$h_c = s \cdot \frac{r}{72} \cdot k \quad (3.10)$$

wobei 72 die Referenz-DPI für Punktgrößen und  $k \approx 0.7$  der typische Verhältnisfaktor zwischen nomineller Schrifthöhe und tatsächlicher x-Höhe (Höhe des Kleinbuchstabens „x“) bezeichnet [53].

Lösungsansätze umfassen:

- **Erhöhte Scanauflösung:** Digitalisierung mit 300–500 DPI statt 150 DPI
- **Interpolationsbasierte Hochskalierung:** Vergrößerung kleiner Bildausschnitte mittels Lanczos-Interpolation vor der OCR-Verarbeitung. Dies ist recheneffizienter als neuronale Super-Resolution und für technische Zeichnungen mit klar definierten Kanten ausreichend.
- **Multi-Scale-Processing:** Verarbeitung auf mehreren Skalierungsstufen mit Ergebnisfusion

### 3.3.6 Qualitätssicherung und Nachbearbeitung

Die Robustheit eines OCR-Systems wird maßgeblich durch die Nachbearbeitungsschritte bestimmt. Für domänenspezifische Anwendungen wie Gleispläne können strukturelle Erwartungen zur Validierung und Korrektur genutzt werden.

#### Konfidenzbasierte Filterung

OCR-Engines liefern typischerweise einen Konfidenzwert  $c \in [0, 1]$  für jedes erkannte Ergebnis. Die Implementierung verwendet kontextabhängige Schwellenwerte:

$$\text{Akzeptanz}(c) = \begin{cases} \text{verwerfen} & c < \tau_{\min} \\ \text{akzeptieren} & c \geq \tau_{\text{final}} \end{cases} \quad (3.11)$$

Dabei werden drei Stufen unterschieden:

- $\tau_{\min} = 0.4$ : Vorfilterung offensichtlich fehlerhafter Detektionen
- $\tau_{\text{early}} = 0.5$ : Early-Exit bei vollständigen, hochkonfidenten Erkennungen zur Effizienzsteigerung
- $\tau_{\text{final}} = 0.8$ : Finale Validierung für sicherheitsrelevante Ausgaben im Bahnbereich

Diese gestufte Filterung ermöglicht eine effiziente Verarbeitung bei gleichzeitiger Minimierung falsch-positiver Erkennungen.

#### Musterbasierte Validierung

Gleisplan-Beschriftungen folgen standardisierten Formaten, die durch reguläre Ausdrücke (Regular Expressions) formalisiert werden können. Ein erkannter Text gilt als valide, wenn er dem erwarteten Muster der jeweiligen Klasse entspricht. Tabelle 3.4 definiert die Validierungsmuster für die wichtigsten Textklassen.

Klasse	Regex-Pattern	Beispiele
Signal	$^{\text{[A-Z]}} \{1,4\} \text{\d}\{1,4\} [\text{a-z}] ?\$$	AS102, BHR201, F3a
Koordinate	$^{\text{\d}\{1,3\}} [.,] \text{\d}\{3,4\} \$$	18.1606, 123,456
GKS	$^{\text{\d}\{4\}} \$$	1234, 5678
Weiche	$^{\text{W}} \text{\d}\{2,4\} [\text{a-z}] ?\$$	W12, W234a

**Tabelle 3.4:** Regex-Validierungsmuster für Gleisplan-Textklassen

### Homoglyphen-Korrektur

OCR-Systeme verwechseln häufig visuell ähnliche Zeichen (Homoglyphen). Tabelle 3.5 listet typische Verwechslungen und deren automatische Korrekturfähigkeit.

Korrekt	Verwechselt mit	Kontextuelle Korrektur	Automatisierbar
O	0, Q	Signal-IDs enthalten keine 0; Koordinaten keine O	Ja
I	1, l	Position im Muster (Anfang: Buchstabe, Mitte: Ziffer)	Teilweise
S	5	Signal-Präfixe sind Buchstaben	Ja
B	8, 3	Kontextabhängig	Teilweise
Z	2	Selten in Signalnamen	Ja

**Tabelle 3.5:** Typische OCR-Homoglyphen und deren Korrekturmöglichkeiten

### Fuzzy-Matching zur Fehlerkorrektur

Bei ungültigen OCR-Erkennungen kann ein Abgleich mit einer Referenzliste bekannter korrekter Werte durchgeführt werden. Die Levenshtein-Distanz [55] quantifiziert dabei die Ähnlichkeit zweier Zeichenketten als minimale Anzahl von Einfüge-, Lösch- und Ersetzungsoperationen zur Transformation eines Strings in einen anderen.

OCR-Ergebnis	Referenz	$d_{\text{Lev}}$	Aktion
AS1O2	AS102	1	Korrektur: O → 0
BHR2O1	BHR201	1	Korrektur: O → 0
XYZAB	AS102	5	Keine Korrektur (zu verschieden)

**Tabelle 3.6:** Fuzzy-Matching mittels Levenshtein-Distanz

Dieses Verfahren wird als *Fuzzy-Matching* bezeichnet, da es unscharfe (approximate) Übereinstimmungen erlaubt. Für sicherheitsrelevante Anwendungen wird ein konservativer Schwellenwert von  $d_{\text{Lev}} \leq 2$  gewählt, um Fehlkorrekturen zu vermeiden. Eine automatische Korrektur wird ausschließlich bei  $d_{\text{Lev}} \leq 2$  durchgeführt, da typische OCR-Fehler (Homoglyphen wie O/0, I/1)

einzelne Zeichenersetzungen darstellen. Höhere Distanzen deuten auf grundlegend fehlerhafte Erkennungen hin, bei denen automatische Korrektur das Risiko von Fehlzuordnungen erhöht.

### 3.3.7 ROI-basierte OCR-Strategien

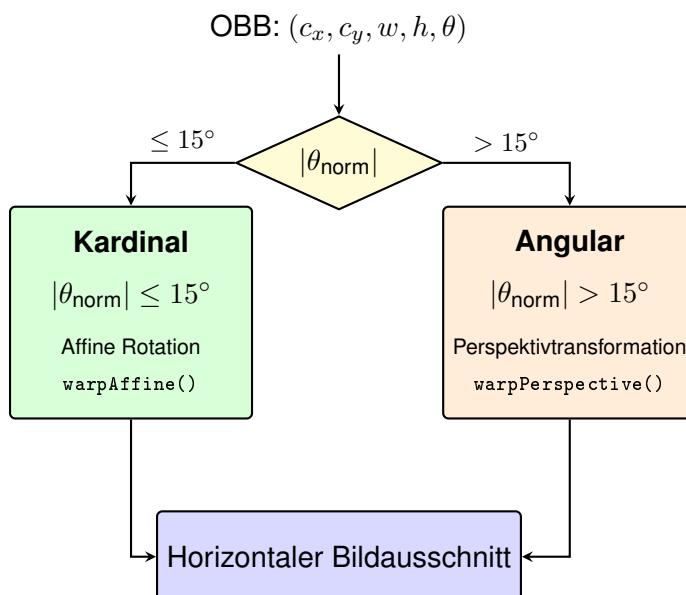
Anstelle einer vollständigen Seitenanalyse (Full-Page OCR) fokussiert der in dieser Arbeit verfolgte Ansatz die OCR auf spezifische Regions of Interest (ROIs), die aus der Objekterkennung abgeleitet werden [56].

#### Vorteile der ROI-fokussierten Verarbeitung

- **Effizienz:** Reduktion des zu verarbeitenden Bildbereichs. Bei den untersuchten Gleisplänen beträgt die kumulierte ROI-Fläche typischerweise unter 5% der Gesamtseitenfläche.
- **Kontextspezifische Verarbeitung:** Klassenabhängige Vorverarbeitungsparameter (z. B. Padding, Skalierung)
- **Reduzierte Falsch-Positive:** Keine Verarbeitung irrelevanter Textbereiche (Legenden, Titelfelder)
- **Geometrische Normalisierung:** Nutzung des OBB-Winkels zur Horizontalausrichtung

#### Orientierungsbewusste ROI-Extraktion

Die Extraktion eines ROI für ein erkanntes Symbol mit orientierter Bounding Box ( $c_x, c_y, w, h, \theta$ ) erfordert eine geometrische Transformation zur Horizontalausrichtung des Textes. Die Implementierung verwendet einen Dual-Pfad-Ansatz, bei dem die Wahl der Transformationsmethode von der Textorientierung abhängt (Abbildung 3.11).



**Abbildung 3.11:** Dual-Pfad ROI-Extraktion: Kardinale Orientierungen verwenden effiziente affine Rotation, während schräge Texte eine Perspektivtransformation erfordern.

**Kardinale Orientierung** ( $|\theta_{\text{norm}}| \leq 15^\circ$ ): Für nahezu achsenparallele Texte genügt eine affine Rotation um den Mittelpunkt  $(c_x, c_y)$ . Die Transformation verwendet eine  $2 \times 3$  Rotationsmatrix:

$$\mathbf{M}_{\text{affin}} = \begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \end{pmatrix} \quad (3.12)$$

wobei  $t_x$  und  $t_y$  die Translationskomponenten bezeichnen, die sicherstellen, dass die Rotation um den Mittelpunkt  $(c_x, c_y)$  erfolgt. Diese Transformation ist recheneffizient und ausreichend für Texte, die bereits nahezu horizontal oder vertikal ausgerichtet sind.

**Beliebige Orientierung** ( $|\theta_{\text{norm}}| > 15^\circ$ ): Für stark geneigte Texte wird eine perspektivische Transformation (Homographie) verwendet [57]. Die vier Eckpunkte  $(x_i, y_i)$  der OBB werden zunächst durch Rotation um den Mittelpunkt berechnet:

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_i - c_x \\ y_i - c_y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \quad (3.13)$$

Anschließend bildet eine  $3 \times 3$  Homographie-Matrix  $\mathbf{H}$  diese Eckpunkte auf ein achsenparalleles Zielrechteck ab:

$$\begin{pmatrix} x''_i \\ y''_i \\ 1 \end{pmatrix} \sim \mathbf{H} \begin{pmatrix} x'_i \\ y'_i \\ 1 \end{pmatrix} \quad (3.14)$$

wobei  $\sim$  Gleichheit bis auf einen Skalierungsfaktor bezeichnet. Die Matrix  $\mathbf{H}$  wird aus den vier korrespondierenden Punktpaaren (Quell-Eckpunkte zu Ziel-Rechteck) berechnet.

Die Entscheidung zwischen beiden Methoden basiert auf dem *normalisierten Winkel*  $\theta_{\text{norm}}$ , der durch die in Abschnitt 6.2.3 beschriebene Normalisierung aus dem Rohwinkel gewonnen wird. Der Schwellenwert von  $15^\circ$  wurde empirisch bestimmt und bietet einen guten Kompromiss zwischen Recheneffizienz (affine Transformation) und geometrischer Genauigkeit (perspektivische Transformation). Bei größeren Abweichungen führt die affine Rotation zu sichtbaren Verzerrungen an den Texträndern, während kleinere Winkel durch die Multi-Rotations-Inferenz ( $0^\circ, 90^\circ, 180^\circ, 270^\circ$ ) ausreichend abgedeckt werden.

### 3.3.8 Relevanz für die Gleisplananalyse

Die beschriebenen OCR-Konzepte bilden die theoretische Grundlage für die in Kapitel 6 (Abschnitt 6.3) beschriebene Implementierung. Die konkrete Umsetzung kombiniert:

- **Multi-Engine-Kaskadierung:** PaddleOCR als primäre Engine aufgrund der integrierten Winkelklassifikation und hohen Geschwindigkeit, mit Tesseract-Fallback für Fälle, in denen PaddleOCR niedrige Konfidenzwerte liefert

- **Dual-Winkel-Routing:** Unterscheidung zwischen kardinaler Rotation ( $0^\circ, 90^\circ, 180^\circ, 270^\circ$ ) und beliebigen Winkeln
- **Klassenspezifische Vorverarbeitung:** Angepasste Padding- und Filterparameter pro Symbolklasse
- **Dreistufige Validierung:** Syntaktisch (Regex), semantisch (Plausibilität), konfidenzbasiert (Schwellenwerte)

Die Texttypen in Gleisplänen umfassen unter anderem Signalbezeichnungen (z. B. „AS102“), Kilometerangaben (z. B. „18.1606“), GKS-Kennungen (z. B. „1234“). Jeder Texttyp erfordert spezifische Validierungsregeln und Nachbearbeitungsstrategien, die in der Implementierung detailliert werden.

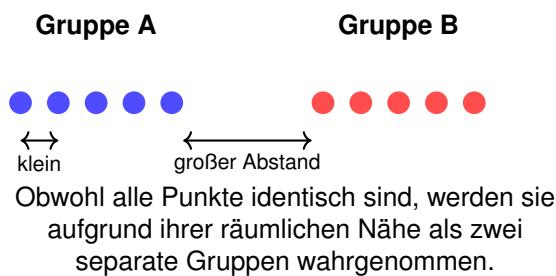
## 3.4 Räumliche Beziehungen in technischen Dokumenten

In den vorangegangenen Abschnitten wurde erläutert, wie einzelne Objekte (Symbole mittels Objekterkennung) und deren textuelle Beschriftungen (mittels OCR) erkannt werden. Die isolierte Erkennung allein genügt jedoch nicht: Ein erkanntes Signalsymbol erhält erst durch die Zuordnung seiner Bezeichnung (z. B. „AS102“) und Kilometrierung (z. B. „18.456“) seine vollständige technische Bedeutung.

Dieser Abschnitt behandelt die theoretischen Grundlagen, wie räumlich getrennte Elemente (Symbole und ihre zugehörigen Texte) automatisch einander zugeordnet werden können.

### 3.4.1 Das Prinzip der räumlichen Nähe

Die automatische Verknüpfung von Elementen basiert auf einem fundamentalen Prinzip der menschlichen Wahrnehmung: dem **Gesetz der Nähe** aus der Gestaltpsychologie [58]. Dieses besagt, dass räumlich nahe beieinander liegende Objekte als zusammengehörig wahrgenommen werden.



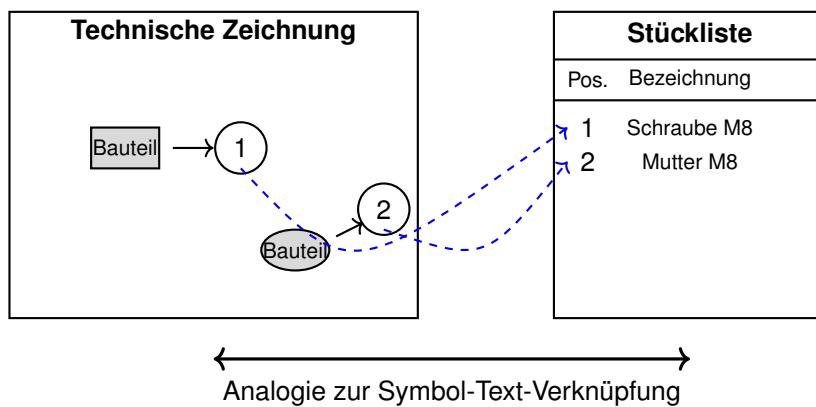
**Abbildung 3.12:** Illustration des Gestaltprinzips der räumlichen Nähe

Dieses psychologische Prinzip wird in der Dokumentenanalyse algorithmisch umgesetzt: Für jedes erkannte Symbol wird der räumlich *nächstgelegene* Text als zugehörig betrachtet. Die

Anwendung von Proximity Heuristiken zur automatischen Layoutanalyse ist ein etabliertes Verfahren in der Dokumentenverarbeitung [59, 60].

### 3.4.2 Analogie zur Stücklistenzuordnung

Für Ingenieure des Maschinenbaus ist das Konzept aus der Arbeit mit technischen Zeichnungen vertraut. Bei einer Explosionsdarstellung mit Positionsnummern ist die Zuordnung zwischen Bauteil und Nummer intuitiv klar: Die Nummer gehört zu dem Bauteil, auf das der Hinweispfeil zeigt oder das sich in räumlicher Nähe befindet (vgl. Abbildung 3.13). Dieselbe Logik liegt der automatischen Symbol-Text-Verknüpfung zugrunde.



**Abbildung 3.13:** Zuordnung von Positionsnummern zu Bauteilen in einer Explosionsdarstellung

### 3.4.3 Herausforderung: Rotierte Elemente

In vielen technischen Zeichnungen folgen Symbole der Geometrie des dargestellten Objekts und können daher in **beliebigen Winkeln** orientiert sein. Dies stellt eine besondere Herausforderung dar: Die Begriffe „oberhalb“, „unterhalb“ oder „rechts von“ verlieren bei rotierten Elementen ihre eindeutige Bedeutung im globalen Koordinatensystem. Abbildung 3.14 illustriert diese Herausforderung: Bei einem horizontal ausgerichteten Symbol ist eindeutig definiert, was „unterhalb“ bedeutet. Bei einem um  $45^\circ$  gedrehten Symbol verliert dieser Begriff jedoch seine eindeutige Bedeutung im globalen Koordinatensystem.



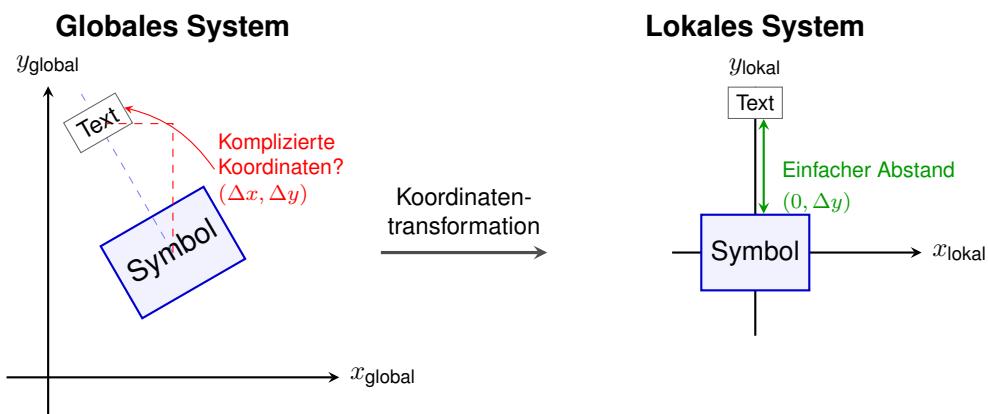
**Abbildung 3.14:** Ambiguität von Richtungsbegriffen bei rotierten Symbolen

### 3.4.4 Lösung: Lokale Koordinatensysteme

Die Lösung besteht darin, für jedes Symbol ein eigenes **lokales Koordinatensystem** zu definieren, das mit der Orientierung des Symbols mitrotiert. In diesem lokalen System haben Richtungsbegriffe wieder eine eindeutige Bedeutung.

Dieses Konzept ist Ingenieuren aus verschiedenen Bereichen vertraut:

- **Technische Mechanik:** Bei der Berechnung von Kräften an einem schräg liegenden Balken transformiert man die Koordinaten in ein lokales System, das mit dem Balken ausgerichtet ist (Hauptachsentransformation) [61].
- **CAD-Systeme:** Beim Platzieren von Features auf geneigten Flächen wird automatisch ein lokales Koordinatensystem (User Coordinate System, UCS) verwendet [10].
- **Robotik:** Werkzeugkoordinatensysteme (Tool Center Point, TCP) rotieren mit dem Endeffektor [62].



**Abbildung 3.15:** Transformation vom globalen ins lokale Koordinatensystem eines Symbols

Wie in Abbildung 3.15 dargestellt, werden durch die Transformation ins lokale System Richtungsbeziehungen wieder eindeutig: Der Begriff „oberhalb“ bezieht sich nun auf die positive  $y_{\text{lokal}}$ -Achse, unabhängig von der globalen Orientierung des Symbols. Die mathematische Umsetzung dieser Transformation, eine Rotation des Koordinatensystems um den Winkel des Symbols, wird in Kapitel 6, Abschnitt 6.4.1 detailliert beschrieben.

### 3.4.5 Klassenspezifische Suchstrategien

In der Praxis variieren die räumlichen Beziehungen zwischen Symbol und Text je nach Objekttyp. Dies spiegelt die Zeichnungskonventionen wider, die sich in verschiedenen Domänen etabliert haben:

Objekttyp	Typische Textposition	Begründung
Signal	Unterhalb des Symbols	Signalbezeichnungen stören nicht die Gleisdarstellung
Kilometerstein	Direkt am Markierungspunkt	Kompakte Darstellung der Streckenposition
Weiche	Variable Positionen	Komplexe Geometrie erfordert flexible Platzierung

**Tabelle 3.7:** Domänenspezifische Konventionen für Textpositionen relativ zu Symbolen

Diese Konventionen werden als **Domänenwissen** in das System integriert und ermöglichen eine gezieltere Suche nach zusammengehörigen Elementen. Derartige domänenspezifische Layoutregeln sind charakteristisch für technische Dokumentationen und folgen etablierten Zeichnungsnormen [11, 63].

## 3.5 Änderungserkennung zwischen Dokumentversionen

Technische Dokumente unterliegen einem kontinuierlichen Änderungsprozess. Im Verlauf eines Projekts werden Pläne mehrfach überarbeitet: Elemente werden hinzugefügt, verschoben oder entfernt; Bezeichnungen werden korrigiert. Die systematische Erfassung dieser Änderungen ist essenziell für die Qualitätssicherung und Nachvollziehbarkeit im Engineering-Prozess.

### 3.5.1 Motivation: Das Revisionswesen

In der industriellen Praxis folgt die Dokumentenlenkung definierten Prozessen gemäß Qualitätsmanagementsystemen wie DIN EN ISO 9001 [64]. Jede Änderung an einer technischen Zeichnung muss:

- **Dokumentiert** werden (Was wurde geändert?)
- **Begründet** werden (Warum wurde geändert?)
- **Freigegeben** werden (Wer hat die Änderung geprüft?)

Die manuelle Identifikation von Änderungen zwischen zwei Planversionen, das sogenannte „Rödeln“ ist zeitaufwändig und fehleranfällig, insbesondere bei umfangreichen Dokumenten mit hunderten von Elementen.

### 3.5.2 Analogie: Stücklistenvergleich

Das Konzept der Änderungserkennung ist Ingenieuren aus dem Stücklistenwesen bekannt. Wenn sich eine Baugruppe ändert, muss die zugehörige Stückliste (Bill of Materials, BOM) aktualisiert werden [65]. Ein BOM-Vergleich identifiziert:

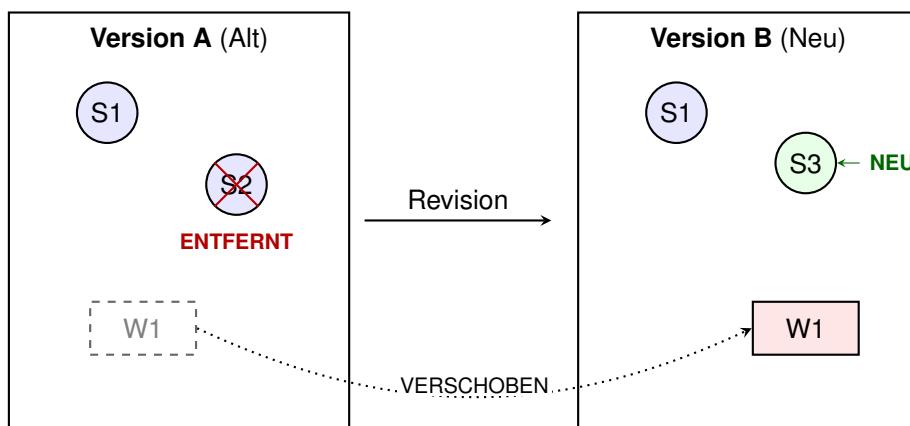
- **Neue Positionen:** Bauteile, die hinzugekommen sind

- **Entfallene Positionen:** Bauteile, die entfernt wurden
- **Verschobene Positionen:** Bauteile an geänderter Einbauposition

Dieselbe Logik lässt sich auf technische Zeichnungen übertragen.

### 3.5.3 Kategorisierung von Änderungen

Abbildung 3.16 zeigt typische Änderungen zwischen zwei Dokumentversionen: Symbol S3 wurde hinzugefügt, Symbol S2 wurde entfernt, und Symbol W1 wurde an eine neue Position verschoben.



**Abbildung 3.16:** Schematische Darstellung von Änderungen zwischen zwei Planversionen.

Änderungstyp	Definition	Praktisches Beispiel
<b>Hinzufügen</b>	Element existiert nur in der neuen Version	Ein zusätzliches Signal wurde installiert
<b>Entfernung</b>	Element existiert nur in der alten Version	Ein Haltepunkt wurde stillgelegt
<b>Verschiebung</b>	Element existiert in beiden Versionen, aber mit geänderter Kilometrierung	Die Streckenposition eines Signals wurde korrigiert

**Tabelle 3.8:** Die drei Grundtypen von Änderungen zwischen Dokumentversionen

### 3.5.4 Prinzip: Identifikation durch eindeutige Merkmale

Um Änderungen automatisch zu erkennen, muss zunächst festgestellt werden, welche Elemente in beiden Versionen „dasselbe“ Objekt darstellen. Dies erfolgt über **eindeutige Identifikationsmerkmale** – vergleichbar mit Artikelnummern in einer Stückliste oder Sachnummern in einem PDM-System [65].

Für technische Zeichnungen können solche Identifikatoren aus den Objektattributen abgeleitet werden. Ein Signal mit der Bezeichnung „AS102“ erhält beispielsweise eine eindeutige Kennung,

die aus Objekttyp und Bezeichnung zusammengesetzt wird. Die algorithmische Umsetzung wird in Kapitel 6, Abschnitt 6.6.1 beschrieben.

### 3.5.5 Das Zuordnungsproblem und der Hungarian Algorithm

Die Kernherausforderung bei der Änderungserkennung besteht in der optimalen Zuordnung von Elementen zwischen zwei Versionen. Bei eindeutigen Identifikatoren (z. B. Signalbezeichnungen) ist die Zuordnung trivial. Problematisch wird es jedoch bei Elementen ohne eindeutige Bezeichner oder bei Duplikaten – mehrere gleichartige Elemente an ähnlichen Positionen.

#### Das Assignment Problem

Formal handelt es sich um ein *Assignment Problem* (Zuordnungsproblem): Gegeben seien zwei Mengen  $A = \{a_1, \dots, a_m\}$  (Elemente der Altversion) und  $B = \{b_1, \dots, b_n\}$  (Elemente der Neuversion) sowie eine Kostenmatrix  $C \in \mathbb{R}^{m \times n}$ , wobei  $c_{ij}$  die „Kosten“ (oder negativen Ähnlichkeitswert) für die Zuordnung von  $a_i$  zu  $b_j$  angibt. Gesucht ist eine Zuordnung, die die Gesamtkosten minimiert [66, 67].

#### Greedy-Ansatz und dessen Limitierungen

Ein naiver Greedy-Ansatz würde für jedes Element der Altversion iterativ den besten noch verfügbaren Partner wählen:

$$\text{match}(a_i) = \arg \max_{b_j \in B_{\text{verfügbar}}} \text{score}(a_i, b_j) \quad (3.15)$$

Dieser Ansatz ist jedoch **nicht optimal**, da frühe Entscheidungen spätere Zuordnungsmöglichkeiten einschränken können. Bei Duplikaten (z. B. zwei Sverbinder an Position km 0.054) können alle Paare ähnliche Scores erhalten, und die reihenfolgeabhängige Greedy-Auswahl führt zu suboptimalen oder falschen Zuordnungen [68].

#### Der Hungarian Algorithm

Der *Hungarian Algorithm*, auch bekannt als *Kuhn-Munkres-Algorithmus*, löst das Assignment Problem in polynomieller Zeit  $\mathcal{O}(n^3)$  und garantiert eine **global optimale** Zuordnung [66, 67].

Der Algorithmus basiert auf dem Konzept der *augmentierenden Pfade* und transformiert die Kostenmatrix iterativ, bis eine optimale Zuordnung gefunden wird. Die wesentlichen Schritte sind:

1. **Matrixreduktion:** Subtraktion des Zeilenminimums von jeder Zeile, dann des Spaltenminimums von jeder Spalte
2. **Nullenüberdeckung:** Finden einer minimalen Anzahl von Linien, die alle Nullen überdecken

3. **Optimierung:** Falls die Anzahl der Linien kleiner als  $n$ , wird die Matrix weiter transformiert
4. **Zuordnung:** Wenn  $n$  Linien benötigt werden, existiert eine optimale Zuordnung über die Nullen

Eigenschaft	Greedy	Hungarian
Zeitkomplexität	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
Optimalität	Lokal	<b>Global</b>
Duplikatbehandlung	Reihenfolgeabhängig	<b>Optimal</b>
Falsche Zuordnungen	Häufig bei Duplikaten	Minimiert

**Tabelle 3.9:** Vergleich von Greedy- und Hungarian-Algorithmus für das Zuordnungsproblem

### Anwendung auf die Änderungserkennung

Für die Änderungserkennung in technischen Zeichnungen wird der Hungarian Algorithm klassenweise angewendet: Für jede Objektklasse (Signale, GKS, Koordinaten, etc.) wird eine separate Kostenmatrix erstellt, wobei die Kosten aus der Ähnlichkeit der Elemente (Textübereinstimmung, räumliche Nähe, Koordinatendifferenz) abgeleitet werden. Zuordnungen mit einem Score unterhalb eines Schwellenwerts werden verworfen – die entsprechenden Elemente gelten als hinzugefügt bzw. entfernt.

## 3.6 Konfigurationsbasierte Systemanpassung

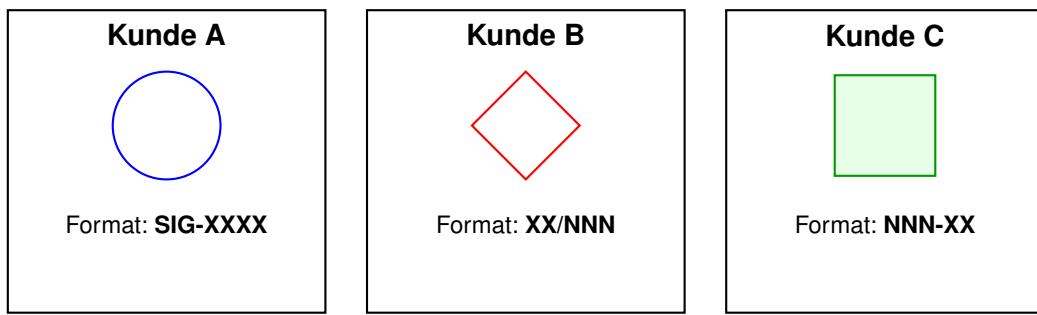
Ein wesentliches Qualitätsmerkmal technischer Software ist ihre **Anpassbarkeit** an unterschiedliche Einsatzszenarien. In der industriellen Praxis variieren Symbolvarianten, Bezeichnungskonventionen und Validierungsregeln zwischen verschiedenen Kunden, Projekten oder Regionen. Eine starre, im Programmcode verankerte Logik würde für jede Anpassung eine Softwareänderung erfordern.

### 3.6.1 Das Problem der Domänenspezifität

Technische Zeichnungen folgen zwar grundlegenden Normen (z. B. DIN für Zeichnungselemente) [11, 63], weisen aber in der Praxis erhebliche Variabilität auf:

- **Kundenspezifische Symbole:** Verschiedene Unternehmen verwenden unterschiedliche Darstellungen für dieselbe technische Funktion
- **Regionale Konventionen:** Bezeichnungsschemata variieren zwischen Ländern
- **Historische Entwicklung:** Ältere Dokumente folgen anderen Standards als aktuelle

Abbildung 3.17 verdeutlicht das Problem: Drei verschiedene Kunden verwenden unterschiedliche Symboldarstellungen und Bezeichnungsformate für dieselbe technische Funktion. Ein starres System müsste für jede Variante angepasst werden.



Wie kann **ein** System alle drei Varianten verarbeiten?

**Abbildung 3.17:** Variabilität von Symboldarstellungen und Bezeichnungskonventionen.

### 3.6.2 Lösung: Trennung von Code und Konfiguration

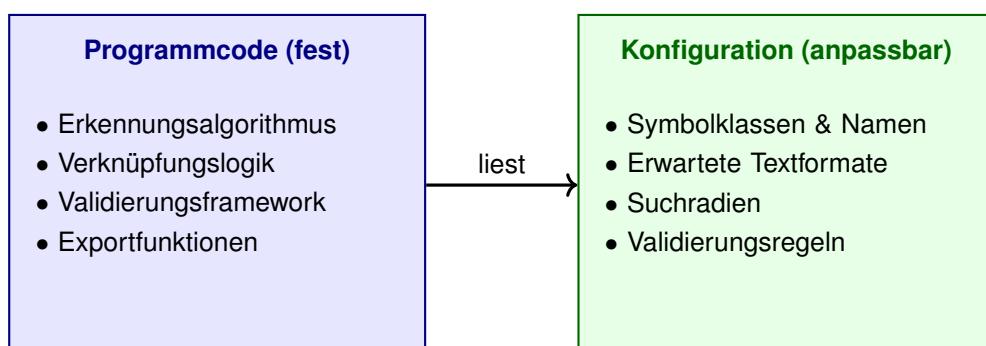
Die Lösung besteht in der strikten Trennung zwischen:

1. **Programmlogik** (Code): Die allgemeinen Algorithmen für Erkennung, Verknüpfung und Validierung, die für alle Anwendungsfälle gleich bleiben
2. **Domänenwissen** (Konfiguration): Die spezifischen Regeln, Parameter und Erwartungswerte, die werden in einem zentralen Konfigurationsmodul definiert

Dieses Prinzip der *Separation of Concerns* ist ein fundamentales Konzept der Softwarearchitektur [69] und ermöglicht die Anpassung des Systems durch Änderung weniger, klar strukturierter Parameter.

Dieses Prinzip ist Ingenieuren aus verschiedenen Bereichen bekannt:

- **Parametrische CAD-Modelle:** Die Geometrie wird durch Parameter gesteuert, nicht durch hartcodierte Maße [70]
- **SPS-Programmierung:** Prozessparameter werden in Datenbausteinen (DBs) abgelegt, nicht im Programmcode [71]
- **FEM-Software:** Materialparameter werden in Bibliotheken gepflegt, nicht im Solver [72]



**Abbildung 3.18:** Architekturprinzip der Trennung von Programmlogik und Konfiguration.

Wie in Abbildung 3.18 dargestellt, liest der unveränderliche Programmcode die domänen spezifischen Parameter aus einem externen Konfigurationsmodul. Anpassungen an neue Kundenanforderungen erfordern dadurch keine Codeänderungen.

### 3.6.3 Typische Konfigurationsparameter

Für ein Dokumentenanalysesystem sind folgende Parameter typischerweise konfigurierbar:

Kategorie	Beispiel	Zweck
Konfidenzschwellen	Klassenspezifische Mindestkonfidenzen	Precisions-/Recall-Balance je Klasse
Verknüpfungsregeln	Suchrichtung, Distanzmultiplikatoren	Steuerung der Symbol-Text-Zuordnung
NMS-Parameter	Klassenspezifische IoU-Schwellen	Unterdrückung von Duplikaten
OCR-Vorverarbeitung	Schärfung, Rauschunterdrückung	Optimierung der Texterkennung

**Tabelle 3.10:** Kategorien konfigurierbarer Parameter

### 3.6.4 Klassenspezifische Parameter

Ein wesentliches Merkmal ist die **klassenspezifische Parametrisierung**: Jede Symbolklasse kann eigene optimierte Werte erhalten, da verschiedene Klassen unterschiedliche Erkennungsschwierigkeiten aufweisen:

- Klassen mit hoher Erkennungssicherheit (z. B. geometrisch eindeutige Symbole) können niedrigere Konfidenzschwellen verwenden
- Klassen mit vielen Falsch-Positiven erfordern strengere Schwellen
- Die Suchrichtung für zugehörigen Text variiert je nach typischer Anordnung im Dokument

### 3.6.5 Orientierungsabhängige Verarbeitung

Zusätzlich zur Klassenspezifik werden Parameter nach **Textorientierung** differenziert. Horizontaler und schräger Text erfordern unterschiedliche OCR-Vorverarbeitung:

- **Kardinale Orientierung** ( $0^\circ, 90^\circ, 180^\circ, 270^\circ$ ): Standardparameter für achsparallelen Text
- **Anguläre Orientierung** (schräg): Angepasste Parameter mit erhöhter Vorverarbeitung

Diese Unterscheidung ermöglicht optimale Erkennungsraten unabhängig von der Textausrichtung im Dokument.

Die konkrete Struktur des Konfigurationsmoduls wird in Kapitel 5 und seine Verarbeitung in Kapitel 6 beschrieben.

## 3.7 Rückverfolgbarkeit und Qualitätssicherung

Bei der automatisierten Extraktion von Daten, insbesondere in sicherheitsrelevanten Domänen, ist die **Rückverfolgbarkeit** der Ergebnisse von fundamentaler Bedeutung. Jeder extrahierte Datensatz muss auf seine Quelle zurückführbar sein, um die Korrektheit überprüfen und bei Fehlern die Ursache identifizieren zu können.

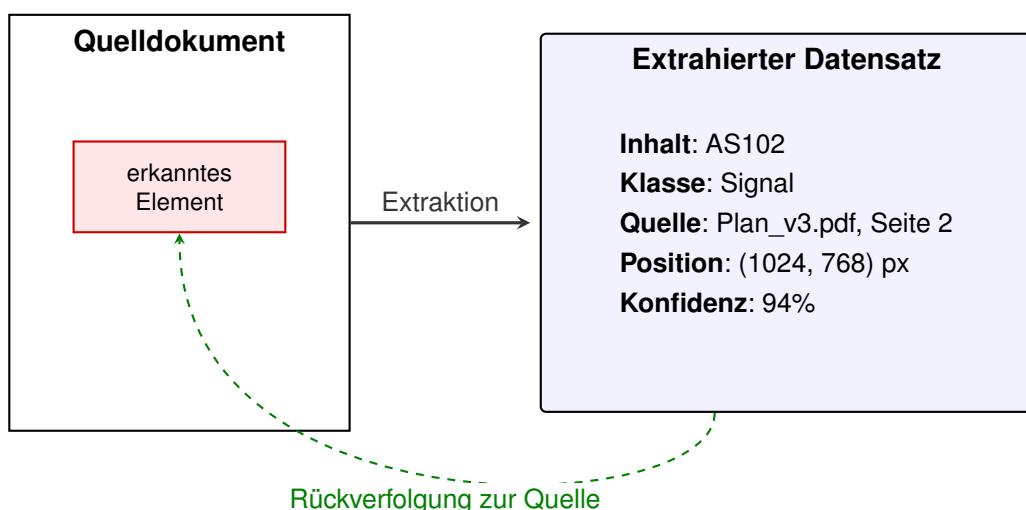
### 3.7.1 Definition und Bedeutung

**Rückverfolgbarkeit** (*Traceability*) bezeichnet die Fähigkeit, für jedes Ergebnis dessen Herkunft und Entstehungsweg nachzuvollziehen [73]. In der Qualitätssicherung ist dieses Prinzip fest verankert – vergleichbar mit:

- **Chargenrückverfolgung** in der Fertigung (jedes Bauteil ist auf seine Herkunft zurückführbar) [74]
- **Audit Trail** in der Prozessindustrie (jede Parameteränderung wird protokolliert) [75]
- **Requirements Tracing** in der Systementwicklung (jede Anforderung ist bis zur Implementierung nachverfolgbar) [76]

### 3.7.2 Komponenten der Rückverfolgbarkeit

Für ein Dokumentenanalysesystem bedeutet Rückverfolgbarkeit, dass zu jedem extrahierten Datensatz folgende Informationen verfügbar sein müssen. Abbildung 3.19 zeigt das Prinzip: Jeder extrahierte Datensatz enthält neben dem eigentlichen Inhalt auch Metadaten zur Quelldatei, Position und Erkennungskonfidenz, die eine Rückverfolgung zur Originalstelle im Dokument ermöglichen.



**Abbildung 3.19:** Struktur eines rückverfolgbaren Datensatzes mit Quellenmetadaten.

Die wesentlichen Metadaten, die für eine vollständige Rückverfolgbarkeit gespeichert werden

müssen, sind in Tabelle 3.11 zusammengefasst.

Information	Beschreibung
Quellenreferenz	Dateiname, Seitennummer, Dokumentversion
Positionsangabe	Koordinaten des erkannten Elements im Originaldokument
Verarbeitungsmetadaten	Verwendete Algorithmen, Zeitstempel der Verarbeitung
Konfidenzinformation	Sicherheit der Erkennung als quantitativer Wert

**Tabelle 3.11:** Wesentliche Metadaten für die Rückverfolgbarkeit

### 3.7.3 Qualitätssicherung durch Validierung

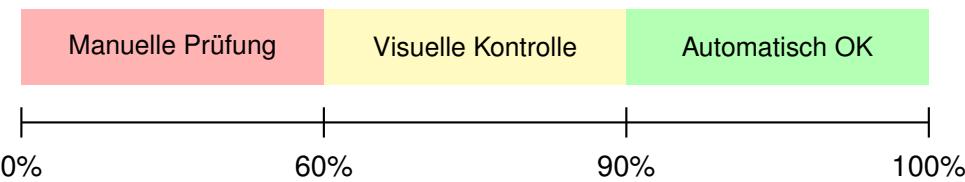
Automatische Erkennungssysteme erreichen nie 100% Genauigkeit. Fehler können in jeder Stufe der Verarbeitungskette auftreten. Um die Auswirkungen solcher Fehler zu minimieren, ist eine **mehrstufige Validierung** erforderlich.

Das Grundprinzip besteht darin, jedes Ergebnis mehreren unabhängigen Plausibilitätsprüfungen zu unterziehen [77]:

- **Formatprüfung:** Entspricht das Ergebnis dem erwarteten syntaktischen Muster?
- **Wertebereichsprüfung:** Liegt der Wert innerhalb plausibler Grenzen?
- **Kontextprüfung:** Ist das Ergebnis im räumlichen/logischen Kontext plausibel?

### 3.7.4 Umgang mit unsicheren Ergebnissen

Moderne Erkennungssysteme liefern zusammen mit dem Ergebnis einen **Konfidenzwert**, der die Sicherheit der Erkennung quantifiziert [78]. Diese Information kann genutzt werden, um Ergebnisse zu klassifizieren. Abbildung 3.20 zeigt eine typische Einteilung: Ergebnisse mit hoher Konfidenz (über 90%) werden automatisch akzeptiert, mittlere Werte erfordern eine visuelle Kontrolle, und niedrige Werte werden zur manuellen Prüfung markiert.



**Abbildung 3.20:** Konfidenzbasierte Klassifikation von Erkennungsergebnissen

### 3.7.5 Human-in-the-Loop: Der Mensch als finale Instanz

Unabhängig von der Leistungsfähigkeit automatischer Systeme bleibt der **Mensch die finale Prüfinstanz**. Dieses Prinzip, in der Informatik als „Human-in-the-Loop“ bezeichnet [79], ist besonders für sicherheitsrelevante Anwendungen unverzichtbar.

Das System sollte daher:

- Unsichere Ergebnisse *markieren*, nicht automatisch verwerfen
- Dem Anwender eine *effiziente Überprüfung* ermöglichen (z. B. durch Navigation zur Quellposition)
- *Korrekturmöglichkeiten* bieten, ohne den Gesamtprozess neu starten zu müssen

Die konkrete Implementierung der Validierungslogik und der Benutzerinteraktion wird in Kapitel 6, Abschnitte 6.5 und 6.7 beschrieben.

### 3.7.6 Zusammenfassung: Theoretische Grundlagen

Die in diesem Kapitel dargestellten theoretischen Konzepte bilden das Fundament für die praktische Implementierung des Extraktionssystems:

- **Objekterkennung** (Abschnitt 3.2): Neuronale Netze für die Detektion von Symbolen
- **Texterkennung** (Abschnitt 3.3): OCR-Verfahren für die Extraktion von Beschriftungen
- **Räumliche Verknüpfung** (Abschnitt 3.4): Proximale Algorithmen mit lokalen Koordinatensystemen
- **Änderungserkennung** (Abschnitt 3.5): Hungarian Algorithm für optimale Elementzuordnung zwischen Dokumentversionen
- **Konfigurierbarkeit** (Abschnitt 3.6): Trennung von Code und Domänenwissen
- **Qualitätssicherung** (Abschnitt 3.7): Rückverfolgbarkeit und mehrstufige Validierung

Die vorgestellten theoretischen Konzepte adressieren gezielt die in Kapitel 2 beschriebene Problemstellung: Während die manuelle Übertragung technischer Pläne in strukturierte Datenformate zeitintensiv, fehleranfällig und kaum skalierbar ist, ermöglichen Deep-Learning-basierte Objekterkennungsverfahren in Kombination mit OCR-Technologien die automatisierte Extraktion von Symbolen und Textinformationen. Die räumliche Verknüpfung dieser Entitäten durch proximale Algorithmen bildet die Grundlage für eine semantische Interpretation, während der Hungarian Algorithm für die Änderungserkennung erstmals eine optimale automatisierte Zuordnung und Verfolgung von Planänderungen ermöglicht. Besonders bedeutsam ist die durchgängige Rückverfolgbarkeit: Durch die Speicherung von Quellenreferenzen und Positionsangaben entsteht eine bidirektionale Verknüpfung zwischen extrahierten Daten und ihrer grafischen Repräsentation im Originaldokument – ein wesentlicher Unterschied zu rein manuellen Workflows, bei denen diese Verknüpfung implizit bleibt und die Validierung erheblich erschwert.

Diese theoretischen Grundlagen schaffen damit die Voraussetzungen für ein hybrides System, das die Stärken automatischer Verfahren (Geschwindigkeit, Konsistenz, Skalierbarkeit) mit menschlicher Expertise (Domänenwissen, Plausibilitätsprüfung) kombiniert. Die konfigurationsbasierte Anpassbarkeit gewährleistet dabei, dass das System auf wechselnde Anforderungen reagieren kann, ohne den Programmcode zu verändern – ein entscheidender Faktor für die

praktische Anwendbarkeit in industriellen Kontexten mit heterogenen Plandarstellungen und kundenspezifischen Symbolbibliotheken.

Die folgenden Kapitel konkretisieren diese theoretischen Grundlagen: Kapitel 4 definiert die funktionalen und nicht-funktionalen Anforderungen, Kapitel 5 beschreibt die Architekturentscheidungen und Kapitel 6 dokumentiert die Implementierung.

## 4. Anforderungsanalyse

Die in Kapitel 3 dargelegten theoretischen und technischen Grundlagen bilden das Fundament für die praktische Umsetzung des Prototyps. In den nachfolgenden Schritten erfolgt die Konkretisierung der Anforderungen an den zu entwickelnden Prototyp. Die vorliegende Analyse leitet sich aus den funktionalen Zielen des Projektes sowie den nicht-funktionalen Rahmenbedingungen im industriellen Umfeld der Siemens Mobility GmbH ab. Zu diesem Zweck werden die Systemgrenzen, Datenformate sowie potenzielle technische Herausforderungen strukturiert analysiert.

### 4.1 Funktionale Anforderungen

Im Rahmen dieser Arbeit wird ein Prototyp entwickelt, der die automatisierte Extraktion und Interpretation von Informationen aus technischen Gleisplänen ermöglicht. Die funktionalen Anforderungen (FA-001 bis FA-014) ergeben sich aus den spezifischen Aufgabenstellungen der Signaltechnik-Planung. Der Prototyp soll folgende Kernfunktionen erfüllen:

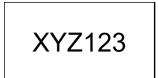
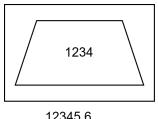
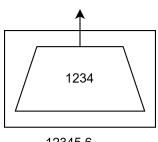
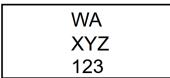
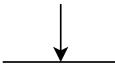
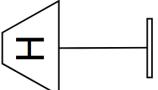
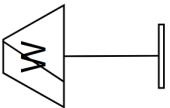
#### 4.1.1 Symbolerkennung und Objektklassifizierung

Eine zentrale Anforderung ist die zuverlässige Detektion bahntechnischer Symbole in Vektor- oder Rastergrafiken.

- **FA-001 Erkennungsrate:** Der Prototyp **muss** mindestens 90 % der definierten Symbolklassen **detektieren** (gemessen als Recall des YOLO-Modells).
- **FA-002 Rotationsinvarianz:** Wenn Symbole in beliebigen Winkeln ( $0^\circ$  bis  $360^\circ$ ) orientiert sind, **muss** der Prototyp in der Lage sein, diese Objekte korrekt zu **klassifizieren** und deren Rotationswinkel mit einer Genauigkeit von  $\pm 5^\circ$  zu **bestimmen**.
- **FA-003 Detektionsziele:** Der Prototyp **muss** 13 Symbolklassen detektieren (Tabelle 4.1). Die Klasse *coordinate* liefert Positionsinformationen (Kilometrierung), die den jeweiligen anderen Klassen zugeordnet werden. Alle 13 Klassen werden in Kapitel 7 quantitativ evaluiert.

#### Symbolklassen

Die folgenden 13 Objektklassen bilden die Detektionsziele des Prototyps. Die Klasse *coordinate* nimmt dabei eine besondere Rolle ein: Sie liefert die Positionsinformation (Kilometrierung), die den anderen 12 Klassen zugeordnet wird:

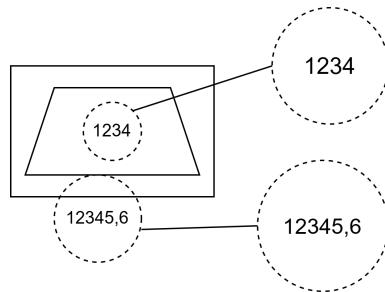
Objektklasse	Symbol
Koordinate (Positionsangabe)	
Signal	
GKS (festkodiert)	
GKS (gesteuert)	
Gleismagnet	
Weichenblock	
Haltepunkt	
Isolierstoß	
S-Verbinder	
Prellbock	
Haltetafel	
Ende Weichen	

Objektklasse	Symbol
Weichengruppenende	

**Tabelle 4.1:** 13 Symbolklassen für die Datenextraktion

#### 4.1.2 Texterkennung und OCR-Integration

- **FA-004 OCR-Genauigkeit:** Der Prototyp **muss** Textinformationen (Signalbezeichnungen, Kilometrierungen) zuverlässig **extrahieren**.
- **FA-005 Robustheit:** Wenn Textregionen unter erschwerten Bedingungen vorliegen (niedriger Kontrast, Bildrauschen, Rotation 0°–360°, Überlagerungen), **muss** der Prototyp in der Lage sein, diese Texte zuverlässig zu **extrahieren**.

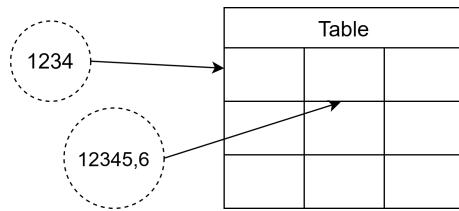
**Abbildung 4.1:** Beispielhafte Extraktion von Text und Position am Symbol GKS

#### 4.1.3 Semantische Verknüpfung und Logik

- **FA-006 Fahrtrichtungsdetektion:** Wenn ein Signal detektiert wird, **soll** der Prototyp die Wirkrichtung (steigend/fallend) basierend auf dem Rotationswinkel des Symbols **ableiten**.
- **FA-007 Symbol-Koordinaten-Verknüpfung:** Der Prototyp **muss** jedes erkannte Gleisplanelement (Signal, GKS, GM-Block) automatisch mit der räumlich nächstgelegenen Kilometrierungsangabe mittels geometrischer Proximity-Analyse unter Berücksichtigung der Symbolorientierung **verknüpfen**.
- **FA-008 Manuelle Korrektur:** Der Prototyp **soll** eine Funktion zur manuellen Überschreibung automatisch erstellter Symbol-Text-Verknüpfungen **bereitstellen**.

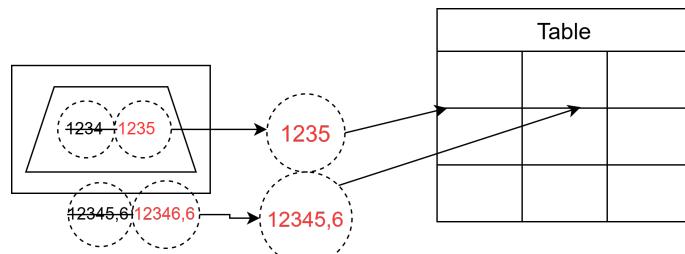
#### 4.1.4 Datenaufbereitung und Export

- **FA-009 Excel-Export:** Der Prototyp **muss** extrahierte Daten vollautomatisch in vordefinierte Excel-Tabellen (.xlsx) mit korrekter Zuordnung zu Zeilen und Spalten **exportieren**.



**Abbildung 4.2:** Schematische Übertragung von erkannten Objektdaten in die Ziel-Tabelle

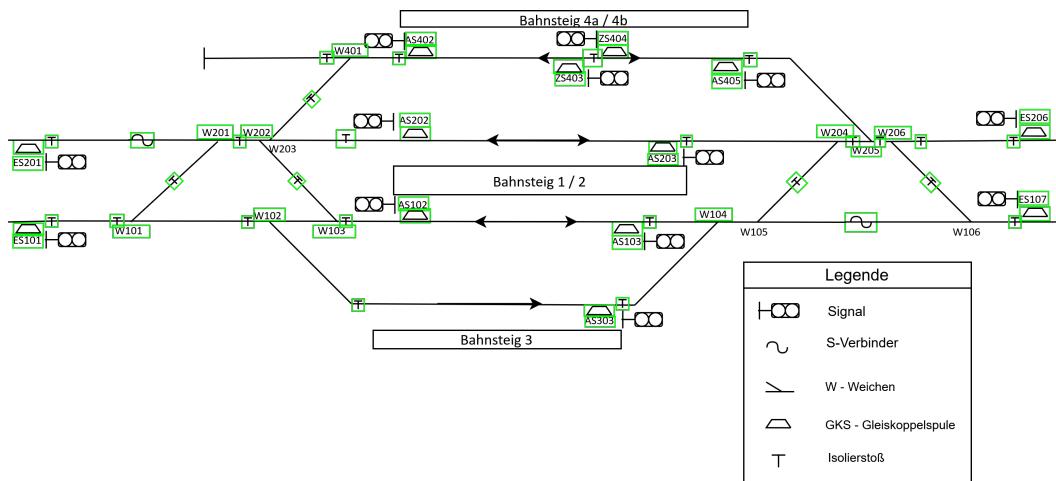
- **FA-010 Strukturerhalt:** Wenn Daten in bestehende Excel-Dateien exportiert werden, **soll** der Prototyp die vorhandene Struktur (Formatierung, Formeln, Makros) **bewahren** und ausschließlich Werte in definierte Bereiche einfügen.
- **FA-011 Änderungsverfolgung:** Wenn zwei Planversionen verglichen werden, **muss** der Prototyp Änderungen **identifizieren** und **kategorisieren**.



**Abbildung 4.3:** Visualisierung der Änderungsverfolgung zwischen zwei Planversionen

#### 4.1.5 Benutzerinteraktion und Konfiguration

- **FA-012 Visuelle Validierung:** Der Prototyp **soll** alle erkannten Objekte durch farbige Bounding Boxes im Gleisplan oder als Overlay zur manuellen Überprüfung **visualisieren**.



**Abbildung 4.4:** Visuelle Validierung durch Bounding-Box-Overlays im Gleisplan [4]

- **FA-013 Grafische Benutzeroberfläche:** Der Prototyp **muss** eine grafische Benutzeroberfläche für PDF-Upload, Analyse-Start und Datenexport ohne erforderliche Kommandozeilen-

Kenntnisse **bereitstellen**.

- **FA-014 Modulare Architektur:** Der Prototyp **muss** mit modularer Architektur **gestaltet sein**, die klare Trennung der Verarbeitungsstufen (Objekterkennung, Texterkennung, UI, Export) gewährleistet und Erweiterungen (z.B. neue Symbolklassen, CAD-Anbindung) ohne Modifikation der Kernlogik ermöglicht.

## 4.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen wurden aus drei komplementären Quellen abgeleitet:

1. **Stakeholder-Anforderungen:** Interviews und Abstimmungen mit dem Betreuer sowie Fachkollegen der Siemens Mobility GmbH identifizierten praxisrelevante Rahmenbedingungen wie On-Premise-Verarbeitung (NFA-001), Lizenzkonformität (NFA-002) und Prozessoptimierung (NFA-006).
2. **Unternehmensrichtlinien:** Die internen IT-Sicherheits- und Compliance-Vorgaben der Siemens Mobility GmbH definieren verbindliche Anforderungen an Datenschutz, Offline-Betrieb und die Verwendung genehmigter Open-Source-Lizenzen.
3. **Domänenanalyse und Stand der Technik:** Aus der Analyse bestehender manueller Arbeitsabläufe (vgl. Kapitel 2, Forschungsvorgehen Phase 2) sowie der Evaluation verwandter Arbeiten (Kapitel 3) wurden Qualitätsanforderungen wie End-to-End-Genauigkeit (NFA-003), Robustheit (NFA-004) und Ressourceneffizienz (NFA-007) abgeleitet.

Ergänzend zu den funktionalen Zielen definieren die nicht-funktionalen Anforderungen (NFA-001 bis NFA-010) die Qualitätsmerkmale und Rahmenbedingungen des Prototyps.

### 4.2.1 Sicherheit und Datenschutz

- **NFA-001 On-Premise-Verarbeitung:** Der Prototyp **muss** alle Daten vollständig lokal ohne Übertragung an externe Cloud-Services **verarbeiten** und Offline-Betrieb **unterstützen**.
- **NFA-002 Lizenzkonformität:** Der Prototyp **muss** ausschließlich Open-Source-Bibliotheken mit genehmigten Lizenzen (Apache 2.0, MIT, BSD) **verwenden** und alle Abhängigkeiten **dokumentieren**.

### 4.2.2 Qualität und Zuverlässigkeit

- **NFA-003 End-to-End-Genauigkeit:** Der Prototyp **muss** eine Gesamtgenauigkeit von mindestens 85 % bei vollständiger Objektextraktion (Detektion + OCR + Linking + Export korrekt) **erreichen**.
- **NFA-004 Robustheit:** Wenn fehlerhafte Eingaben vorliegen (korrupte PDFs, Bildrauschen), **soll** der Prototyp definierte Fehlermeldungen statt Absturz **bereitstellen**.

- **NFA-005 Prüfbarkeit:** Der Prototyp **soll** visuelle Validierung aller Ergebnisse mittels Bounding-Box-Overlay und Rückverfolgbarkeit (Klick auf Tabellenzeile zeigt Position im Plan) **bereitstellen**.

#### 4.2.3 Effizienz und Wirtschaftlichkeit

- **NFA-006 Prozessoptimierung:** Der Prototyp **soll** den manuellen Prüfaufwand durch Transformation vom 4-Augen-Prinzip hin zu einem KI-gestützten Prozess (Mensch prüft KI) **reduzieren**.
- **NFA-007 Ressourceneffizienz:** Der Prototyp **soll** einen durchschnittlichen Bahnhofsplan auf Standard-Hardware (CPU-only) in weniger als der Hälfte der manuellen Bearbeitungszeit **verarbeiten**.

#### 4.2.4 Wartbarkeit und Erweiterbarkeit

- **NFA-008 Update-Fähigkeit:** Der Prototyp **soll** die Integration neuer Symbolvarianten über Konfigurations-Updates oder neue Modell-Gewichte ohne Code-Änderungen **unterstützen**.

#### 4.2.5 Datenformate und Schnittstellen

- **NFA-009 Eingabeformate:** Der Prototyp **muss** PDF-Dateien (Vektor/Raster, intern gerendert bei 500 DPI) und Bilddateien (PNG, JPEG, TIFF, BMP mit nativer Auflösung von 500 DPI) **verarbeiten**.

*Begründung:* Das YOLOv8-OBB Modell wurde ausschließlich auf 500 DPI Bildkacheln trainiert. Abweichungen führen zu signifikant reduzierten Erkennungsraten.

- **NFA-010 Ausgabeformate:** Ergänzend zum Excel-Export (FA-009) **soll** der Prototyp Ergebnisse auch in CSV und JSON Format **exportieren**.

Tabelle 4.2 gibt einen detaillierten Überblick über die im Prototyp verwendeten Dateiformate und deren Einsatz in der Verarbeitungspipeline.

Kategorie	Format	Beschreibung	Einsatz im Prototyp
<b>Eingabe</b>	PDF	Standardformat für Pläne	Primäre Datenquelle
	PNG/JPG	Rasterisierte Ausschnitte	Input für CNN/YOLO
<b>Verarbeitung</b>	JSON	Strukturierte Metadaten	Interner Datenaustausch
	PostgreSQL	Relationale Datenbank	Persistenz & Versionierung
<b>Ausgabe</b>	XLSX	Excel-Arbeitsmappe	Engineering-Workflow
	CSV	Textbasiertes Format	Einfacher Datenaustausch
	JSON	API-Response	Schnittstellenanbindung

**Tabelle 4.2:** Übersicht der unterstützten Datenformate (siehe 4.2.5 und 4.2.5)

## 4.3 Herausforderungen bei der Umsetzung

Die Realisierung der in den Abschnitten 4.1.1 bis 4.2.5 definierten Anforderungen sieht sich folgenden technischen Herausforderungen gegenüber:

1. **Daten-Heterogenität:** Die Varianz in den Eingabedaten (unterschiedliche Export-Einstellungen, Linienstärken, Skalierungen) erschwert eine universelle Regelbildung.
2. **Visuelle Ambiguität:** Einige Symbole (z. B. unterschiedliche Gleiskoppelpulsen-Typen) unterscheiden sich visuell nur in wenigen Pixeln oder sind nur durch den Kontext (Begleittext) differenzierbar.
3. **OCR-Komplexität:** Technischer Text in Plänen ist oft extrem klein, rotiert und durch Führungslinien durchgestrichen, was klassische OCR-Engines (wie Tesseract) an ihre Grenzen bringt.
4. **Mangel an Trainingsdaten:** Es existiert kein öffentlicher Datensatz für bahntechnische Symbolik. Ein „Cold Start“ ist notwendig, bei dem Trainingsdaten zunächst manuell (z. B. via CVAT) annotiert werden müssen.
5. **Semantische Lücke:** Der Schritt von der Erkennung („Da ist eine Box“) zur Bedeutung („Das ist Weiche 12 in Rechtslage“) erfordert komplexe Heuristiken, insbesondere beim Mapping von Textboxen zu den geometrisch nächsten Symbolen.

## 4.4 Anforderungs-Rückverfolgbarkeit

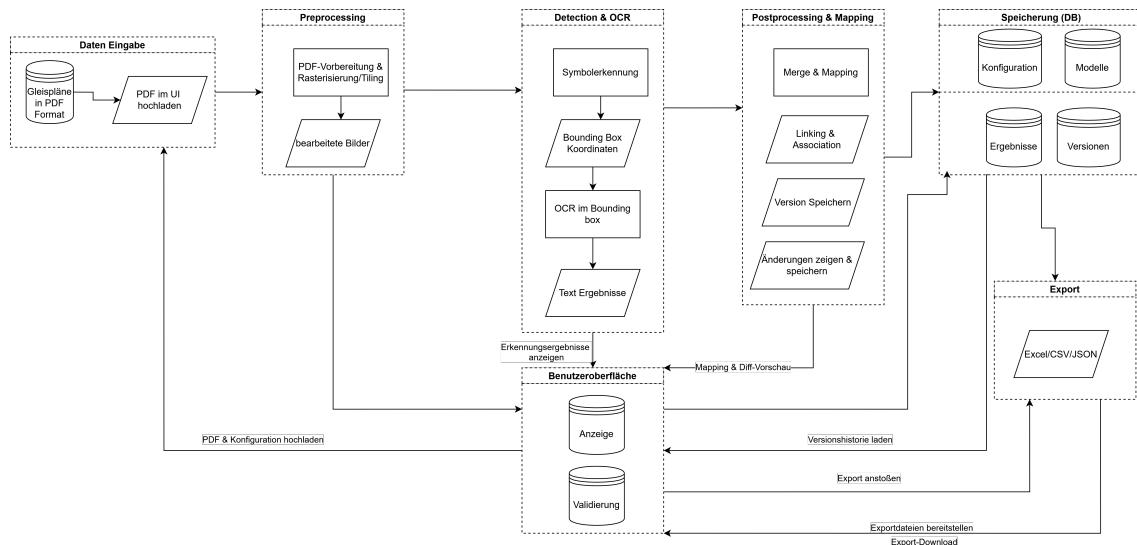
Die vollständige Rückverfolgbarkeitsmatrix, welche die Zuordnung jeder Anforderung zu den entsprechenden Implementierungskomponenten sowie den Evaluationsmetriken dokumentiert, wird in Kapitel 7, Abschnitt 7.3 präsentiert.

# 5. Konzeption des Prototyps

In diesem Kapitel wird der Entwurf und die technische Konzeption des Prototyps aufbauend auf der im vorherigen Kapitel definierten Anforderungsanalyse dargelegt. Das Ziel besteht in der Entwicklung einer modularen und skalierbaren Prototyparchitektur, die den gesamten Workflow von der rohen PDF-Datei bis zum strukturierten und validierten Datenexport automatisiert. Die konkrete Implementierungsdetails werden anschließend in Kapitel 6 behandelt.

## 5.1 Prototyparchitektur

Die Architektur des Prototyps folgt dem Prinzip der Modularität und Separation of Concerns. Der Prototyp ist in funktional abgegrenzte Module unterteilt, die über definierte Schnittstellen miteinander kommunizieren. Diese Architektur ermöglicht eine schrittweise Entwicklung, einfache Wartbarkeit und die Möglichkeit zur späteren Erweiterung um zusätzliche Funktionalitäten (**Anforderungen FA-014, NFA-008**). Abbildung 5.1 zeigt die Gesamtarchitektur des Prototyps mit ihren Hauptkomponenten.



**Abbildung 5.1:** Modulare Prototyparchitektur des Gleisplanextraktors

### 5.1.1 Dateneingabe

Der Prototyp akzeptiert Gleispläne in verschiedenen Formaten als primäre Eingabe (**Anforderung NFA-009**). Die Unterstützung mehrerer Formate berücksichtigt sowohl den aktuellen Anwendungsfall als auch die zukünftige Erweiterbarkeit auf andere Gleisplan-Layouts.

#### Unterstützte Eingabeformate:

- **PDF-Dateien (.pdf):** Standardformat für technische Zeichnungen. Der Prototyp rendert PDFs intern bei einer festen Auflösung von 500 DPI, was eine konsistente und optimale

Bildqualität gewährleistet.

- **Rastergrafiken (.png, .jpg/.jpeg, .tif/.tiff, .bmp):** Für bereits rasterisierte Pläne aus Scans oder Bildexporten. Diese werden mit ihrer nativen Auflösung verarbeitet.

### Auflösungsanforderung für das aktuelle Layout:

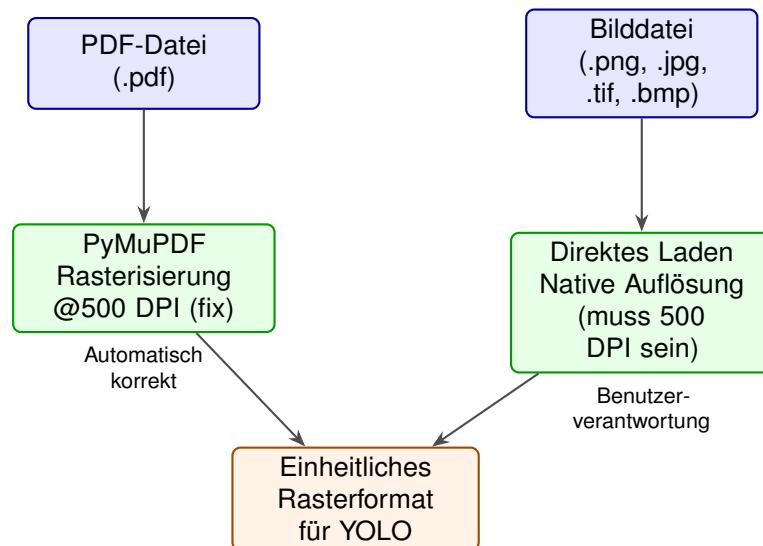
Für das in dieser Arbeit behandelte Gleisplan-Layout ist eine Auflösung von exakt 500 DPI **zwingend erforderlich**. Diese strikte Anforderung ergibt sich aus dem Trainingsverfahren des YOLOv8-OBB Modells: Sämtliche Trainingsdaten wurden aus bei 500 DPI gerenderten PDFs extrahiert (vgl. Abschnitt 6.2.1). Das Modell hat somit Pixelmuster gelernt, die spezifisch für diese Auflösung sind. Bei abweichenden Auflösungen, selbst bei 300 oder 400 DPI, stimmen die Symbolgrößen und Detailstrukturen nicht mehr mit den gelernten Mustern überein, was zu erheblichen Erkennungsfehlern führt.

Bei PDF-Eingaben wird die korrekte Auflösung durch die interne Rasterisierung automatisch sichergestellt. Bei Bilddateien muss der Benutzer gewährleisten, dass die Eingabe mit 500 DPI erstellt wurde. Der Prototyp zeigt einen entsprechenden Hinweis in der Benutzeroberfläche an.

### Erweiterbarkeit für zukünftige Layouts:

Die Unterstützung von Bildformaten neben PDF dient primär der zukünftigen Erweiterbarkeit. Für neue Gleisplan-Layouts mit anderen grafischen Konventionen könnte ein separates YOLO-Modell mit angepasster Trainingsauflösung entwickelt werden. Die flexible Eingabearchitektur ermöglicht dies ohne Änderungen am grundlegenden Systemaufbau.

Abbildung 5.2 visualisiert die differenzierte Behandlung der beiden Eingabepfade.



**Abbildung 5.2:** Differenzierte Eingabeverarbeitung für PDF- und Bilddateien.

### DPI-Konsistenz und Modellkompatibilität:

Da das YOLOv8-OBB Modell auf Bildausschnitten trainiert wurde, die aus bei 500 DPI ge-

renderten PDFs stammen, ist die Auflösung der Eingabedaten ein kritischer Faktor für die Erkennungsgenauigkeit. Bei PDF-Eingaben wird dies durch die interne Rasterisierung automatisch sichergestellt. Bei Bilddateien liegt die Verantwortung für eine ausreichende Auflösung beim Benutzer. Der Prototyp zeigt daher bei Bildformaten einen Hinweis auf die empfohlene Mindestauflösung an:

- **Optimal (500 DPI):** Volle Übereinstimmung mit Trainingsdaten, maximale Erkennungsgenauigkeit
- **Akzeptabel (300–499 DPI):** Geringfügig reduzierte Genauigkeit, für die meisten Anwendungsfälle ausreichend
- **Kritisch (< 300 DPI):** Deutlich erhöhte Fehlerrate, nicht empfohlen

### 5.1.2 Vorverarbeitung

Die Vorverarbeitung transformiert die PDF Dokumente in ein für die nachfolgenden KI Module geeignetes Format. Dieser Schritt ist essentiell, weil Deep Learning basierte Objekterkennungsmodelle pixelbasierte Eingaben benötigen. Kernaufgaben des Vorverarbeitungsmoduls:

1. **Rasterisierung:** Konvertierung der PDF Seiten in hochauflösende Bilddateien (z.B. PNG, JPEG). Die Auflösung muss ausreichend hoch gewählt werden, um auch kleine Symbole und Beschriftungen erkennbar zu machen, typischerweise 300 DPI.
2. **Normalisierung:** Einheitliche Farbkanalbehandlung (z.B. Konvertierung zu Graustufen) zur Reduktion der Datenkomplexität bei gleichzeitiger Beibehaltung der relevanten visuellen Information.
3. **Tiling (Kachelbildung):** Große Gleispläne überschreiten häufig die maximale Eingabegröße moderner Objekterkennung (typischerweise 640x640 bis 1024x1024 Pixel). Daher wird das Gesamtbild in überlappende Ausschnitte (Tiles) unterteilt. Die Überlappung stellt sicher, dass Symbole an Kachelgrenzen nicht unvollständig erfasst werden.

Designentscheidungen: Die Größe der Tiles und der Überlappungsgrad sind konfigurierbare Parameter, die je nach Plandichte und Symbolgrößen angepasst werden können.

#### Formatunabhängige Weiterverarbeitung:

Unabhängig vom Eingabeformat (PDF oder Bild) sind alle nachfolgenden Verarbeitungsschritte (Tiling, YOLO-Inferenz, OCR und Linking) identisch. Dies wird durch die einheitliche Repräsentation als RGB-Array nach der Eingabeverarbeitung gewährleistet (vgl. Abbildung 6.8 in Kapitel 6). Die Tile-Größe von  $2048 \times 2048$  Pixeln und der Überlappungsgrad von 12,5% bleiben konstant, wobei die absolute Anzahl der Tiles von den Bildabmessungen und damit indirekt von der Eingabeauflösung abhängt.

#### Differenzierte Behandlung nach Eingabeformat:

Die Vorverarbeitung unterscheidet zwischen PDF- und Bildeingaben:

1. **PDF-Eingaben:** Rasterisierung mittels PyMuPDF bei fester Auflösung von 500 DPI. Die Transformationsmatrix gewährleistet eine deterministische und reproduzierbare Bildqualität unabhängig von der Original-PDF-Auflösung.
2. **Bildeingaben (PNG, JPEG, TIFF, BMP):** Direkte Verwendung der nativen Pixeldaten ohne Reskalierung. Dies erhält die Originalqualität, erfordert aber eine angemessene Eingabeauflösung durch den Benutzer. Eine Hochskalierung niedrig aufgelöster Bilder wird bewusst vermieden, da sie keine neuen Informationen generiert und zu Artefakten führen kann.

Die nachfolgenden Verarbeitungsschritte (Tiling, Normalisierung) sind unabhängig vom Eingabeformat identisch, wodurch eine konsistente Pipeline für alle unterstützten Datentypen gewährleistet wird.

### 5.1.3 Detection & OCR

Dieses Modul bildet das technologische Herzstück des Prototyps und kombiniert zwei komplementäre Erkennungsverfahren:

1. **Objekterkennung** - Ein trainiertes Deep Learning Modell der YOLO Familie analysiert die vorverarbeiteten Bildausschnitte und identifiziert relevante Symbole (Signale, Weichen, GKS, etc.). Für jedes erkannte Objekt liefert das Modell:
  - Die Klasse (z.B. „Signal“, „Weiche“)
  - Die Bounding Box (räumliche Position im Bild)
  - Einen Konfidenzwert (0-1), der die Sicherheit der Erkennung ergibt.

Die Verwendung von Oriented Bounding Boxes (OBB) ist konzeptionell vorgesehen, um rotierte Symbole präziser zu erfassen, was eine häufige Herausforderung in technischen Zeichnungen ist.

2. **Texterkennung** - OCR Verfahren extrahieren textuelle Informationen aus den erkannten Symbolbereichen. Die Texterkennung erfolgt räumlich fokussiert innerhalb oder in definierter Nähe der durch die Objekterkennung identifizierten Bounding Boxes. Dieser zielgerichtete Ansatz reduziert Fehler durch irrelevante Textfragmente im Gleisplan.  
**Konzeptionelle Herausforderung:** Text in technischen Zeichnungen kann in verschiedenen Orientierungen vorliegen. Das Konzept sieht daher eine orientierungsbewusste OCR Strategie vor, bei der Textregionen vor Erkennung normalisiert werden.

### 5.1.4 Nachbearbeitung und Mapping

Die Rohausgaben der Erkennungsmodule erfordern weitere Verarbeitungsschritte, um strukturierte und semantisch korrekte Daten zu erzeugen:

1. **Datenbereinigung:** Rohe OCR Ergebnisse enthalten häufig Artefakte, Formatierungsfehler oder unerwünschte Zeichen. Regelbasierte Filter (z.B. mittels parametrischer Beziehungen) normalisieren die extrahierten Texte gemäß erwarteten Mustern.
2. **Zusammenführung:** Die im Schritt erzeugten überlappenden Kacheln müssen wieder zu einem Gesamtbild zusammengefügt werden. Dabei werden Duplikate in den Überlappungsbereichen durch Non Maximum Suppression (NMS) eliminiert [80].
3. **Semantisches Mapping:** Die erkannten Symbole und Texte werden in einen semantischen Kontext überführt. Beispielsweise wird ein detektiertes Symbol der Klasse „Signal“ mit dem dazugehörigen OCR Text zu einer logischen Einheit verknüpft.

### 5.1.5 Speicherung

Alle extrahierten und verarbeiteten Daten werden in einem strukturierten Persistenzsystem gespeichert:

#### Konzeptionelle Anforderungen:

- **Versionierung:** Jede Analyse eines Gleisplans erhält eine eindeutige Versions ID, um spätere Vergleiche zu ermöglichen.
- **Nachverfolgbarkeit:** Jedem Datensatz werden Metadaten zugeordnet (Quelldokument, Seitennummer, Koordinaten), um die Rückverfolgbarkeit zur Originalquelle zu gewährleisten.
- **Flexibles Schema:** Verwendung eines Datenbanksystems, das sowohl strukturierte als auch semi strukturierte Daten effizient speichern kann (z.B. PostgreSQL mit JSONB-Support)

### 5.1.6 Benutzeroberfläche

Die UI dient als Kontrollzentrum für alle Systemfunktionen und ermöglicht die Interaktion ohne Programmierkenntnisse:

#### Konzeptionelle Anforderungen:

- **Prozessvisualisierung:** Anzeige des Verarbeitungsfortschritts in Echtzeit.
- **Ergebnisvalidierung:** Visuelle Darstellung der erkannten Objekte überlagert auf dem originalen Gleisplan.
- **Konfigurationszugriff:** Bearbeitung von Systemparametern über intuitive Eingabemasken
- **Fehlerbehandlung:** Klare Fehlermeldungen und Hilfestellungen bei Problemen

### 5.1.7 Export

Das Exportmodul transformiert die interne Datenrepräsentation in kundengerechte Ausgabeformate:

#### Unterstützte Formate:

- Excel(.xlsx): Strukturierte Tabellen für direkte Integration in bestehende Engineering Workflows
- CSV: Austauschformat für einfache Datenweiterverarbeitung
- JSON: Maschinenlesbares Format für API Integration in nachgelagerte Systeme

**Konzeptionelle Anforderung:** Das Exportmodul muss konfigurierbare Templates unterstützen, um unterschiedliche Kundenvorgaben für Tabellenstrukturen abzubilden.

## 5.2 Workflow: Von PDF zu Excel

Der zu entwickelnde Prototyp basiert auf einer klaren Struktur eines Swimlane-Diagramms mit drei Ebenen, wodurch eine saubere Trennung zwischen Benutzeroberfläche, Datenverarbeitung und Datenspeicherung gewährleistet wird. Das folgende Flussdiagramm 5.3 veranschaulicht den gesamten Prozess von der ersten Benutzereingabe bis zum endgültigen Datenexport.

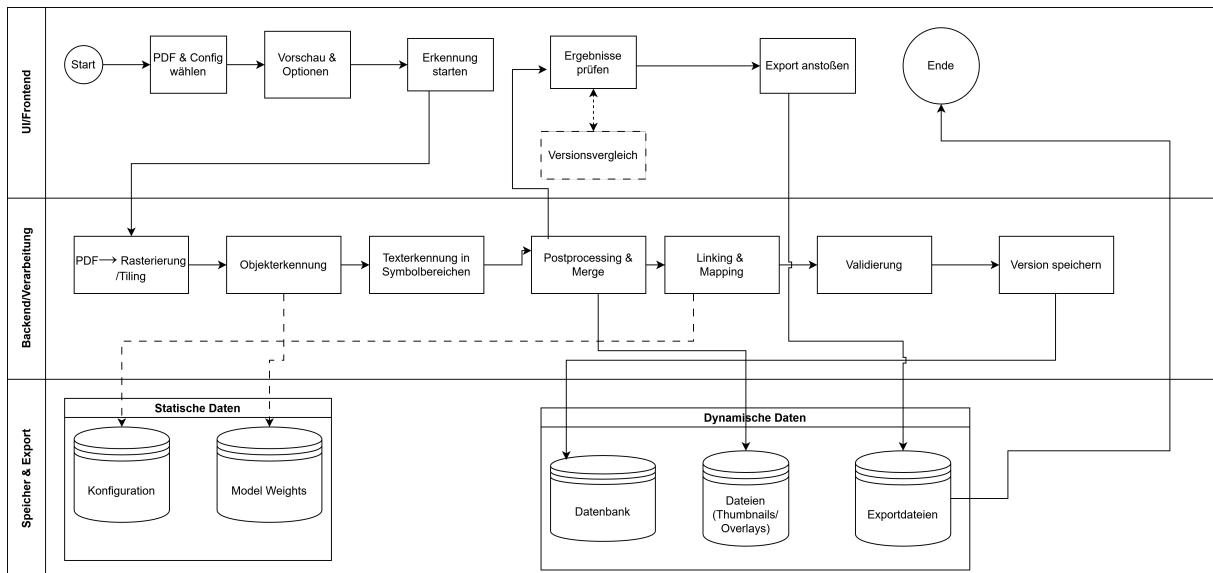


Abbildung 5.3: Workflowdiagramm des Prototyps

### 5.2.1 1. Ebene - UI/Frontend (Benutzeroberfläche)

Die Frontend-Ebene bildet die Schnittstelle zwischen Anwender und System. Der konzeptionelle Ablauf umfasst:

1. **Dateiauswahl:** Der Benutzer wählt eine PDF-Datei sowie optional eine Konfigurationsdatei aus

2. **Parametereinstellung:** Konfiguration von Verarbeitungsoptionen (z.B. zu analysierende Seiten, Erkennungsmodell, OCR Engine)
3. **Prozessinitiierung:** Start der automatisierten Analyse durch Betätigung einer „Ausführen“-Schaltfläche
4. **Ergebnisanzeige:** Darstellung der Erkennungsergebnisse zur manuellen Überprüfung
5. **Export:** Finalisierung und Download der aufbereiteten Daten im gewünschten Format

Designprinzip: Der Workflow folgt einem linearen Assistenten-Modell (Wizard Pattern), das den Benutzer schrittweise durch den Prozess führt.

### 5.2.2 2. Ebene - Backend/Verarbeitung (Kernlogik)

Das Backend orchestriert die automatisierte Analysepipeline. Die Prozessschritte sind:

1. **Rasterisierung:** Konvertierung der PDF-Seiten in Pixelbilder hoher Auflösung
2. **Tiling:** Segmentierung in überlappende Bildausschnitte für effiziente Verarbeitung
3. **Objekterkennung:** Detektion relevanter Symbole mittels Deep Learning Modell
4. **OCR:** Extraktion von Textinformationen aus identifizierten Symbolbereichen
5. **Postprocessing & Merge:** Zusammenführung der Tile-Ergebnisse zu einem Gesamtbild und Eliminierung von Duplikaten in Überlappungsbereichen durch Non-Maximum-Suppression (NMS) [80]. Rohe OCR-Ergebnisse werden durch regelbasierte Filter bereinigt.
6. **Linking & Semantisches Mapping:** Verknüpfung erkannter Symbole mit zugehörigen Texten (IDs, Koordinaten) zu logischen Einheiten. Beispielsweise wird ein detektiertes Symbol der Klasse „Signal“ mit dem dazugehörigen OCR-Text zu einer semantisch interpretierbaren Entität verbunden. Die konzeptionellen Strategien hierfür werden in Abschnitt 5.4 detailliert beschrieben.
7. **Validierung & Qualitätskontrolle:** Automatisierte dreistufige Plausibilitätsprüfung der Erkennungsergebnisse auf Symbol-, OCR- und Linking-Ebene. Fehlgeschlagene Validierungen werden zur manuellen Überprüfung markiert (Human-in-the-Loop). Das vollständige Validierungskonzept ist in Abschnitt 5.5 dargelegt.
8. **Versionsspeicherung:** Persistierung der Ergebnisse mit Versionsmetadaten in der Datenbank. Jede Analyse erhält eine eindeutige Versions-ID, um spätere Vergleiche zu ermöglichen.
9. **Exportvorbereitung:** Transformation der internen Datenrepräsentation in das gewünschte AusgabefORMAT

Designprinzip: Die Pipeline folgt dem Pipes-and-Filters Architekturmuster, bei dem jede Komponente eine spezifische Transformation durchführt und das Ergebnis an die nächste Stufe weiterreicht.

**Optionaler Versionsvergleich:** Unabhängig von der automatischen Verarbeitungspipeline kann der Benutzer über die Benutzeroberfläche einen manuellen Vergleich zwischen zwei Versionen desselben Gleisplans initiieren. Hierzu werden beide Planversionen in separaten Ansichten (Tabs) geladen. Durch Betätigung einer Vergleichsfunktion analysiert der Prototyp beide Datensätze und identifiziert Änderungen (hinzugefügte, entfernte oder modifizierte Objekte). Die konzeptionelle Vergleichsstrategie basiert auf dem in Abschnitt 5.3.6 beschriebenen hybriden Matching-Ansatz. Die Ergebnisse werden visuell hervorgehoben dargestellt, sodass der Benutzer Planänderungen nachvollziehen kann. Die technische Implementierung wird in Kapitel 6 beschrieben.

### 5.2.3 3. Ebene - Speicher und Export (Datenverwaltung)

Die unterste Ebene verwaltet die Daten des gesamten Systems, die in statischen und dynamischen Daten unterteilt sind:

- **Statische Daten** umfassen die Basiskonfigurationsdaten, die die Regeln für das Mapping und die Verarbeitungsparameter definieren, sowie die trainierten Gewichte des Erkennungsmodells. Diese Daten dienen als feste Eingabeparameter für die Verarbeitung im Backend und ändern sich während der Laufzeit nicht.
- **Dynamische Daten** werden während des Verarbeitungsprozesses generiert. Zu diesem Zweck ist eine zentrale Datenbank für die dauerhafte Speicherung der Analyseergebnisse und Versionsstände erforderlich. Darüber hinaus werden temporäre Dateien wie Anzeigebilder und Overlays zwischengespeichert. Die aufbereiteten Daten werden schließlich in Form von Exportdateien bereitgestellt, die dem Nutzer zum Download zur Verfügung stehen.

Designprinzip: Die Trennung zwischen statischen und dynamischen Daten folgt dem Konzept der Immutability für Konfigurationen, während Analyseergebnisse in einem transaktionalen Datenbanksystem versioniert gespeichert werden.

*Hinweis:* Die konkrete technologische Umsetzung (Datenbankschema, Dateiformate, Speicherorganisation) wird in Kapitel 6 detailliert beschrieben.

## 5.3 Designentscheidungen

Die Entwicklung des Prototyps erforderte fundamentale Entscheidungen bezüglich der eingesetzten Technologien und Architekturansätze. Die nachfolgenden Abschnitte dokumentieren diese Designentscheidungen mit technischer Begründung und Abgrenzung zu alternativen Lösungsansätzen.

### 5.3.1 Datenaufbereitung: PDF-Rasterisierung mit pdf2image

Da moderne Objekterkennungsmodelle (wie YOLO) auf Pixeldaten operieren, müssen die vektorbasierten PDF-Gleispläne in einem ersten Schritt rasterisiert werden. Hierfür wurde die Bibliothek `pdf2image` gewählt, die als Python-Wrapper für die C++-Bibliothek *Poppler* fungiert.[81]

Diese Komponente bildet den kritischen Eingangskanal der Pipeline. Die Entscheidung für `pdf2image` begründet sich durch folgende technische Eigenschaften:

- **Rendering-Qualität (Cairo Engine):** Poppler nutzt intern die *Cairo*-Rendering-Engine, die eine pixelgenaue Rasterisierung von Vektorlinien und eingebetteten Textelementen gewährleistet. Für die vorliegende Arbeit wurde eine Auflösung von **500 DPI** gewählt. Diese hohe Pixeldichte ist notwendig, um filigrane Symbole (z. B. Weichenzungen oder Sperrsignale) auch nach der Kachelung (Tiling) noch differenzierbar für das neuronale Netz darzustellen [82].
- **Speichereffizientes Ressourcenmanagement:** Gleispläne weisen extreme Dimensionen auf (bis zu  $67.000 \times 7.000$  Pixel). Ein naives Laden solcher Dateien würde den Arbeitsspeicher (RAM) handelsüblicher Workstations überlasten. `pdf2image` unterstützt eine Streaming-Verarbeitung, bei der Seiten einzeln in den Speicher geladen und sofort verarbeitet werden, ohne das Gesamtdokument im RAM zu halten.
- **Nahtlose Python-Integration:** Die Bibliothek liefert die gerenderten Daten direkt als *PIL*-Objekte (Python Imaging Library) zurück. Dies ermöglicht eine In-Memory-Weitergabe an den *NumPy/OpenCV*-Stack für das Preprocessing, ohne den I/O-Flaschenhals des Zwischenspeicherns auf der Festplatte [81].

#### Abgrenzung zu alternativen Bibliotheken

Die Auswahl wurde gegen populäre Alternativen validiert. Die folgende Tabelle fasst die Ausschlusskriterien zusammen, wobei insbesondere lizenzerrechtliche Aspekte im Unternehmenskontext eine Rolle spielten:

Alternative	Nachteil / Ausschlussgrund
<b>PyMuPDF (fitz)</b>	<i>Lizenzrisiko:</i> Obwohl PyMuPDF performant ist, unterliegt es der <b>AGPL-Lizenz</b> [83]. Für eine kommerzielle Nutzung innerhalb von Siemens Mobility würde dies eine Offenlegung des gesamten Quellcodes erzwingen (Copyleft-Effekt), was für interne Tools oft ausgeschlossen ist.
<b>PDFminer</b>	<i>Falscher Fokus:</i> PDFminer extrahiert die Dokumentenstruktur (Text, Metadaten), ist aber nicht in der Lage, komplexe Vektorgrafiken visuell korrekt zu rasterisieren. Für visuelle Objekterkennung ist es daher ungeeignet [84].
<b>Poppler CLI</b>	<i>Systembruch:</i> Die direkte Nutzung der Kommandozeilen-Tools ( <code>pdftoppm</code> ) erfordert Systemaufrufe via <code>subprocess</code> . Dies erschwert das Exception-Handling und verhindert den direkten Datenaustausch im RAM (erzwungener File-I/O), was die Pipeline verlangsamt.

**Tabelle 5.1:** Vergleich verschiedener PDF-Verarbeitungsbibliotheken

Die konkrete Wahl der Rendering-Parameter (DPI-Auflösung, Farbtiefe, Anti-Aliasing) und die Implementierung der Pipeline wird in Abschnitt 6.2.3 (Preprocessing und Rasterisierung) beschrieben.

### 5.3.2 Objekterkennung: Einsatz von YOLOv8 OBB

Zur Identifizierung der schematischen Symbole (Signale, Weichen, Gleiselemente) wurde das Modell **YOLOv8** (You Only Look Once) in der spezialisierten Konfiguration für rotierte Bounding Boxes (OBB) implementiert. Diese Wahl stellt eine Abkehr von klassischen Detektoren dar und wird durch die spezifische Topologie von Gleisplänen begründet.

Die Entscheidung für YOLOv8 OBB basiert auf drei technischen Hauptfaktoren[22]:

1. **Rotationsinvarianz durch OBB-Regression:** Technische Gleispläne zeichnen sich durch eine hohe Dichte an Symbolen aus, die entlang der Gleisachsen in beliebigen Winkeln ( $\theta \in [-90^\circ, +90^\circ]$ ) angeordnet sind. Herkömmliche Detektoren mit achsenparallelen Boxen (Horizontal Bounding Boxes, HBB) sind hier ungeeignet, da sie:

- Bei diagonalen Objekten (z. B.  $45^\circ$ ) einen hohen Anteil an irrelevanter Hintergrundfläche einschließen (geringe Signal-to-Noise Ratio).
- Bei dichter Symbolfolge zu signifikanten Überlappungen (Intersection over Union, IoU) führen, was fälschlicherweise die Non-Maximum-Suppression (NMS) auslöst und benachbarte Objekte unterdrückt.

YOLOv8 OBB löst dieses Problem durch die Vorhersage eines 5-dimensionalen Vektors  $(c_x, c_y, w, h, \theta)$ , wobei  $\theta$  den Rotationswinkel beschreibt. Dies ermöglicht eine präzise Umhüllung der Symbole unabhängig von ihrer Orientierung im Plan [85].

- 2. Echtzeitfähigkeit und Tiling-Effizienz:** Da Gleispläne oft Auflösungen von über 50.000 Pixeln Breite aufweisen, müssen sie in Hunderte kleinerer Kacheln (Tiles) zerlegt werden. Die Inferenzgeschwindigkeit ist daher kritisch für die Gesamtaufzeit. YOLOv8 erreicht auf Standard-GPU-Instanzen (z.B. AWS g4dn.xlarge) eine Framerate von  $\sim 40$  FPS bei einer Auflösung von  $1024 \times 1024$ . Dies ermöglicht die Verarbeitung eines gesamten Bahnhofsplans in wenigen Sekunden, was für eine interaktive Nutzung der Anwendung essenziell ist [22].
- 3. Architektonische Modularität (Anchor-Free):** Im Gegensatz zu älteren YOLO-Versionen nutzt v8 einen *Anchor-Free*-Ansatz. Dies eliminiert die Notwendigkeit, vor dem Training manuelle Anker-Boxen basierend auf der Verteilung der Symbolgrößen zu berechnen (K-Means Clustering). Dies reduziert das Hyperparameter-Tuning erheblich und verbessert die Generalisierung auf neue, bisher unbekannte Symbolklassen [86].

### Abgrenzung zu alternativen Detektionsverfahren

Im Auswahlprozess wurden verschiedene Architekturen evaluiert. Die folgende Tabelle verdeutlicht, warum diese trotz spezifischer Stärken für den Anwendungsfall „Gleisplan“ verworfen wurden:

Alternative	Evaluationsergebnis	Grund für die Ablehnung
<b>RTMDet</b>	Hohe Präzision (mAP: 91.2%), vergleichbar mit YOLOv8	Obwohl die Rotation-Aware-Features vielversprechend sind, zeigte RTMDet eine instabilere Konvergenz bei kleinen Datensätzen ( $< 2000$ Samples) und verfügt über ein kleineres Entwickler-Ökosystem im Vergleich zu YOLO [87].
<b>Template Matching (OpenCV)</b>	Geringer Recall ( $< 60\%$ ) bei leichten Rotationen ( $\pm 10^\circ$ )	Klassische Computer-Vision-Verfahren sind nicht robust gegenüber Skalierung und Artefakten. Zudem ist der Ansatz bei $> 13$ Symbolklassen rechnerisch nicht skalierbar [88].
<b>DETR / DINOv2</b>	Trainingszeit ca. $3\times$ länger als bei CNNs	Transformer-basierte Ansätze benötigen enorme Datenmengen, um die Inductive Biases von CNNs zu lernen. Für die strukturierte Domäne technischer Zeichnungen stellt dies einen unverhältnismäßigen Ressourcenaufwand dar [89].
<b>Faster R-CNN</b>	Inferenz: $\sim 8$ FPS (vs. 40 FPS bei YOLO)	Die Two-Stage-Architektur (Region Proposal Network + Classifier) ist zwar präzise, aber für das Feedback in einer User-Interface-Anwendung zu langsam [20].

**Tabelle 5.2:** Vergleich und Bewertung von Alternativen zu YOLOv8

Die Entscheidung für YOLOv8-OBB adressiert direkt die Anforderungen **FA-001** (Erkennungsrate  $\geq 90\%$ ), **FA-002** (Rotationsinvarianz durch OBB-Regression) und **NFA-007** (Ressourceneffizienz durch Echtzeitfähigkeit).

### 5.3.3 Texterkennung: Multi-Engine-Strategie und Dual-Angle-Routing

Die Extraktion von Textinformationen aus technischen Gleisplänen stellt aufgrund variierender Schriftarten, unterschiedlicher Orientierungen ( $0^\circ, 90^\circ, 270^\circ$ ) und der Überlagerung durch grafische Elemente eine besondere Herausforderung dar. Um eine maximale Robustheit zu gewährleisten, wurde keine einfache OCR-Lösung gewählt, sondern eine **kaskadierte Multi-Engine-Strategie** implementiert.

Der Prototyp kombiniert die Deep-Learning-basierte Bibliothek **PaddleOCR** (als Primärinstanz) mit der klassischen Engine **Tesseract** (als Fallback). Ergänzt wird dies durch einen eigens entwickelten **Dual-Angle-Routing-Algorithmus**.

#### Primäre Engine: PaddleOCR (PP-OCRv3)

Als Hauptkomponente kommt PaddleOCR zum Einsatz. Die Entscheidung für dieses Framework basiert auf der Architektur des PP-OCRv3-Modells, welches auf einem ultra-leichten CRNN (Convolutional Recurrent Neural Network) basiert.[90]

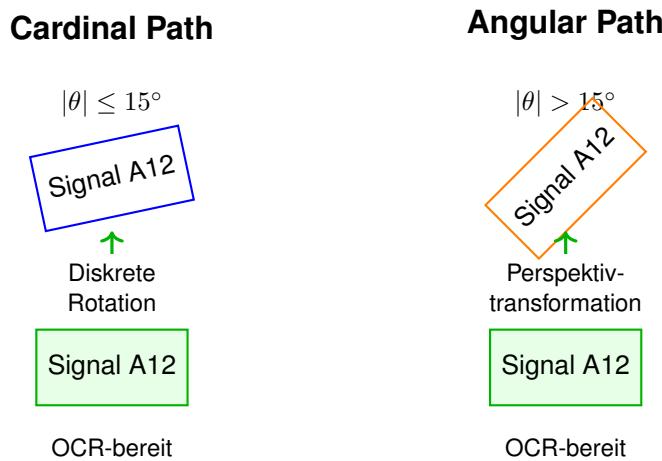
Die technischen Vorteile gegenüber alternativen Engines zeigten sich in der empirischen Evaluation:

- **Rotation-Robustheit:** Standard-OCR-Engines versagen oft bei vertikalem Text. PaddleOCR unterstützt zwar eine interne Klassifizierung (`use_angle_cls`), diese erwies sich jedoch bei kurzen technischen Bezeichnern als fehleranfällig. Daher wurde der Prototyp so konfiguriert, dass die Rotation extern durch die OBB-Koordinaten (aus dem YOLO-Schritt) vorgegeben wird, was die Stabilität signifikant erhöht [46].
- **Genauigkeit (Character Error Rate - CER):** Auf einem validierten Testset von Gleisplan-Ausschnitten (Crops) erzielte PaddleOCR die niedrigsten Fehlerraten:
  - **PaddleOCR:** 7.2 % (horizontal), 12.1 % (vertikal/rotiert)
  - **Tesseract:** 11.4 % (horizontal), 23.8 % (vertikal/rotiert)
  - **EasyOCR:** 9.1 % (horizontal), 18.3 % (vertikal/rotiert)
- **Inferenzgeschwindigkeit:** Mit durchschnittlich **52 ms pro Crop** (CPU, 12 Threads) ist PaddleOCR deutlich effizienter als Tesseract (78 ms), was bei Plänen mit tausenden Textobjekten essentiell für die Gesamlaufzeit ist.

### Algorithmus: Dual Angle Routing

Ein neuartiger Ansatz der Pipeline ist das *Dual Angle Routing*. Da die Orientierung eines Textes im CAD-Plan nicht immer eindeutig aus der Bounding Box hervorgeht (z. B. bei quadratischen Boxen), wird jeder Textausschnitt spekulativ in zwei Orientierungen verarbeitet:

1. **Pfad A (Original):** Der Crop wird unverändert an die OCR übergeben.
2. **Pfad B (Orthogonal):** Der Crop wird um  $90^\circ$  rotiert an die OCR übergeben.



**Abbildung 5.4:** Konzeptioneller Vergleich der Dual-Winkel-Verarbeitungsstrategien: Cardinal Path für nahezu ausgerichtete Texte ( $|\theta| \leq 15^\circ$ ) mit diskreter Rotation; Angular Path für stark geneigte Texte ( $|\theta| > 15^\circ$ ) mit Perspektivtransformation

Die Entscheidung, welches Ergebnis übernommen wird, erfolgt durch eine **Textnormierungs-Heuristik**. Beide Ergebnisse werden gegen reguläre Ausdrücke (Regex) für typische Signalbezeichnungen (z. B.  $^{\wedge} [A-Z] \{1,3\} [0-9]^{+}$ ) geprüft. Das Ergebnis mit dem höheren Konfidenz-Score, das gleichzeitig dem Schema entspricht, wird akzeptiert. Dies eliminiert Rauschen, das entsteht, wenn vertikaler Text fälschlicherweise horizontal gelesen wird.

### Fallback-Ebene und Vorverarbeitung

Für Fälle, in denen PaddleOCR keine plausiblen Ergebnisse liefert ( $\text{Konfidenz} < 0.5$ ), greift der Prototyp auf **Tesseract 5** zurück.

- **Spezialisierung:** Tesseract zeigt insbesondere bei stark pixelierten, aber hochkontrastigen Signalnamen Stärken.
- **Page Segmentation Modes (PSM):** Durch die explizite Setzung von `-psm 7` (Single Line) für Koordinaten und `-psm 8` (Single Word) für IDs wird die Engine gezwungen, Layout-Analysen zu überspringen, was Fehlinterpretationen reduziert [47].
- **Preprocessing:** Vor der OCR-Anwendung durchlaufen die Bildausschnitte eine adaptive Binarisierung nach Otsu [91], um Artefakte der PDF-Rasterung zu entfernen und den Zeichenkontrast zu maximieren.

### Abgrenzung zu Alternativen

Die folgende Tabelle fasst zusammen, warum aktuelle Transformer-basierte Ansätze („State of the Art“ in der Forschung) für diesen spezifischen Anwendungsfall nicht geeignet waren:

Alternative	Grund für den Ausschluss
LayoutLMv3	<i>Ressourcen-Overhead</i> : Als Transformer-Modell benötigt es GPU-Beschleunigung und ca. 2.1 GB VRAM. Für die Erkennung einzelner kurzer Textzeilen (Single-Line OCR) stellt dies einen unverhältnismäßigen Ressourcenaufwand („Overkill“) dar [92].
Donut (End-to-End)	<i>Mangelnde Generalisierung</i> : Da Donut (Document Understanding Transformer) direkt Bild-zu-Token generiert, ohne explizite Bounding Boxes zu nutzen, scheitert es oft an den ungewöhnlichen Vektorschriftarten der CAD-Pläne, da kein spezifisches Vortraining existiert [93].
TrOCR	<i>Rotations-Schwäche</i> : TrOCR ist auf horizontale Textzeilen spezialisiert. Da Gleispläne Text in beliebigen Winkeln enthalten, wäre ein komplexes Pre-Alignment notwendig, was den Vorteil der Architektur negiert [44].

**Tabelle 5.3:** Vergleich und Bewertung alternativer OCR-Ansätze

Die Multi-Engine-Strategie mit Dual-Angle-Routing erfüllt die Anforderungen **FA-004** (OCR-Genauigkeit) und **FA-005** (OCR-Robustheit bei Rotation und Rauschen).

#### 5.3.4 Datenexport und Reporting: Excel Schnittstelle

Für den Export der extrahierten Informationen und die Übergabe an die Fachabteilung wird die Funktion `pandas.DataFrame.to_excel()` in Kombination mit der `openpyxl`-Engine verwendet. Diese Architektur ermöglicht die direkte Serialisierung der internen Datenstrukturen in das weit verbreitete `.xlsx`-Format (Office Open XML).[94]

Die Entscheidung für diesen Ansatz stützt sich auf drei zentrale Gründe:

1. **Feature-Vollständigkeit und Usability**: Die Lösung bietet native Unterstützung für semantische Formatierungen, die die manuelle Weiterverarbeitung erleichtern:
  - **Visuelle Strukturierung**: Anpassung der Spaltenbreite, Festlegen von Zellenfarben und Fixierung von Kopfzeilen (Freeze Panes).
  - **Rückverfolgbarkeit (Traceability)**: Verwendung von Formeln wie `=HYPERLINK()`, um vom Datensatz direkt auf den Bildausschnitt im Plan zu verweisen.
  - **Multi-Sheet-Architektur**: Erstellung mehrerer Arbeitsblätter (Tabs) je nach Datenklasse (z. B. separate Blätter für Signale und Weichen).
2. **Integration mit pandas**: Es entsteht eine nahtlose Datenfluss-Pipeline ohne Medienbruch. Die im Analyse-Schritt erzeugten DataFrames werden direkt persistiert, wobei `openpyxl`

als Backend dient, um auch bestehende Templates modifizieren zu können.

### 3. Unternehmenskonformität:

- Das .xlsx-Format ist der Industriestandard bei Siemens Mobility.
- Die Dateien sind kompatibel mit Microsoft Excel ( $\geq 2016$ ) und LibreOffice Calc.
- Da keine Makros (.xlsm) benötigt werden, entfallen sicherheitstechnische Hürden beim Austausch.

Im Rahmen der Architekturentscheidung wurden folgende Alternativen evaluiert und verworfen:

Alternative	Nachteil / Ausschlussgrund
XlsxWriter	Reine Write-Only-Bibliothek [95]. Sie ist zwar performant, kann aber existierende .xlsx-Dateien nicht einlesen oder bearbeiten, was die Nutzung von Vorlagen (Templates) verhindert.
CSV (Textdatei)	Keine Unterstützung für Formatierungen, Formeln oder mehrere Arbeitsblätter. Zudem bestehen Risiken durch Encoding-Probleme (UTF-8 vs. ANSI) und Trennzeichen-Konflikte.
LibreOffice API / COM	Systemabhängig und langsam. Erfordert eine lokale Installation der Office-Suite auf dem Server, was die Portabilität der Anwendung einschränkt.

**Tabelle 5.4:** Vergleich der Export-Alternativen zu pandas + openpyxl

Die Implementierung der Export-Pipeline, einschließlich der Tabellenformatierung, bedingten Formatierung und Formelintegration, wird in Abschnitt 6.7.7 (Excel-Export-Modul) beschrieben. Diese Architekturentscheidung erfüllt die Anforderungen **FA-009** (Excel-Integration), **FA-010** (Strukturerhalt bei bestehendem Template) und **NFA-010** (Ausgabeformate).

### 5.3.5 Rückverfolgbarkeit: Koordinaten + Bildausschnitte

Die gewählte Lösung für die Rückverfolgbarkeit ist ein hybrider Ansatz mit Koordinaten in Excel und optionaler Miniaturbildgenerierung. Für die UI-Interaktion wurde eine einfache Methode verwendet: Wird die Tabellenzeile in der Benutzeroberfläche angeklickt, so wird dem Benutzer ein vergrößerter Bereich von  $2048 \times 2048$  px im Layout angezeigt, wobei die entsprechenden Datenüberlagerungen in leuchtendem Rot hervorgehoben sind.

Die Gründe für diese gewählte Lösung sind:

- Kompaktheit: Die Koordinaten benötigen nur 8 Byte und 50 KB pro Miniaturansicht, was einen sehr geringen Speicherplatzbedarf darstellt und eine sehr hohe Funktionalität bieten.
- Flexibilität: Auf Wunsch des Benutzers kann die Benutzeroberfläche den entsprechenden vergrößerten Bereich für eine genauere und präzisere Analyse darstellen.

- Normkonformität: Diese Methode erfüllt außerdem die Anforderungen der VDI 1000:2017-06 hinsichtlich Rückverfolgbarkeit.[96]

Andere Alternativen:

Alternative	Nachteil
<b>Nur Bildausschnitte</b>	Datenmenge: 12 MB für 250 Symbole; keine maschinelle Weiterverarbeitung
<b>Nur Hyperlinks</b>	Externe PDF-Abhängigkeit; kein Inline-Preview in UI
<b>Bildoverlay in Excel</b>	Excel Dateigröße >100 MB; Leistungsprobleme

**Tabelle 5.5:** Alternativen zur hybriden Rückverfolgbarkeit

### 5.3.6 Änderungsverfolgung: Konzeptioneller Ansatz

Der Prototyp unterstützt die Versionierung von Gleisplänen, um Änderungen zwischen verschiedenen Planständen nachvollziehbar zu machen.

#### Konzeptionelle Strategie

Die Änderungserkennung basiert auf einem zweistufigen Vergleichsansatz:

1. **Objektidentifikation:** Jedes extrahierte Symbol erhält eine eindeutige Kennung (UID), die aus Objektklasse und Textinhalt gebildet wird. Dies ermöglicht die eindeutige Zuordnung zwischen Planversionen.

Beispiel: Ein Signal mit der Bezeichnung „AS102“ erhält die UID `signal_AS102`.

2. **Differenzbildung:** Mittels Hungarian Algorithm werden Elemente optimal zugeordnet und Änderungen kategorisiert:

- **Hinzugefügt:** Objekte, die nur in der neuen Version existieren
- **Entfernt:** Objekte, die nur in der alten Version existieren
- **Verschoben:** Objekte mit geänderter Kilometrierung (`coord_value`)

#### Optimale Zuordnung mittels Hungarian Algorithm

Für die Zuordnung von Elementen zwischen zwei Planversionen wird der Hungarian Algorithm eingesetzt, der eine global optimale Zuweisung garantiert. Dies ist besonders wichtig bei:

- **Duplikaten:** Mehrere gleichartige Elemente an ähnlichen Positionen
- **Fehlenden Bezeichnern:** Symbol-Klassen ohne eindeutige Textkennung

Die Ergebnisse werden als strukturierter Änderungsbericht exportiert, der die Art der Änderung, die betroffenen Attribute und die Positionen dokumentiert.

Die technische Implementierung dieser Strategie, einschließlich der mathematischen Formulierung und der Algorithmen, wird in Abschnitt 6.6.1 detailliert beschrieben.

### 5.3.7 Benutzeroberfläche: PyQt5

Für die Benutzeroberfläche wurde das Framework PyQt5 aus der Python-Bibliothek ausgewählt. Die Gründe für diese Wahl werden nachfolgend dargelegt[97]:

1. Leistung und Rendering Architektur - Die Kernaufgabe der Anwendung ist die Visualisierung von extrem hochauflösenden Gleisplänen (bis zu 67.000 x 7.000 Pixel).
  - Hardwarebeschleunigung: PyQt5 nutzt über die QGraphicsView-Klasse direkt die OpenGL-Schnittstelle der Grafikkarte. Dies ermöglicht eine Hardwarebeschleunigung für Zoomen und Schwenken ohne Ruckeln.
  - Multithreading: Um das Blockieren der Benutzeroberfläche (UI Freezing) während rechenintensiver Operationen (YOLO-Inferenz, OCR-Verarbeitung) zu verhindern, wurde eine asynchrone Architektur mittels QThread und dem Signal-Slot-Mechanismus implementiert. Dies garantiert, dass die GUI auch während der Analyse reaktionsfähig bleibt.
2. Unternehmenssicherheit und Datensouveränität: Im Zusammenhang mit kritischer Eisenbahninfrastruktur ist der Schutz vertrauenswürdiger Gleisplanungsdaten von höchster Priorität:
  - On-Premise Deployment: Die Anwendung läuft vollständig lokal auf dem Rechner des Ingenieurs. Es findet keine Datenübertragung an externe Server oder Cloud-Dienste statt.
  - Security by Design: Durch den Verzicht auf Browser-Technologien entfallen typische Web-Angriffsvektoren wie Cross-Site Scripting (XSS) oder unerwünschtes Data-Leakage durch Browser-Extensions. Die Datenhoheit bleibt im lokalen Dateisystem.
3. Ergonomie und Multi-Monitor-Workflows: PyQt5 zeichnet sich durch eine hohe Flexibilität in der Fensterverwaltung aus. Ingenieure nutzen in vielen Fällen mehrere Monitore, um den Gleisplan auf einem Bildschirm und die tabellarischen Daten auf dem anderen Bildschirm zur Überprüfung anzuzeigen. Mittels dieses Frameworks ist es möglich, die Fenster derart zu arrangieren, dass die Daten im Gleisplan effizient aufgefunden und überprüft werden können. Diese Lösung ermöglicht zudem das dynamische Ausklappen (Popping out) von Widgets. Dies trägt zur Ergonomie bei und reduziert den Zeit- und Arbeitsaufwand beim Wechseln zwischen Fenstern oder Registerkarten.
4. Vergleich mit alternativen Technologien: Im Vorfeld der Entwicklung wurden verschiedene UI-Frameworks evaluiert. Die folgende Tabelle fasst die Gründe für deren Ausschluss zusammen:

Alternative	Technologie	Hauptgrund für den Ausschluss
Streamlit	Python Web-Wrapper	Kein Multi-Window Support- Statelessness erzwingt ständiges Neuladen (Rerun) bei Interaktionen, was bei großen Bildern zu inakzeptablen Wartezeiten führt.[98]
Flask / Django	Web Backend	Overhead & Komplexität [99]- Erfordert lokalen Webserver und Browser; keine native Integration in das Dateisystem (Drag & Drop, native Dialoge).
Electron	JS/Chromium	Ressourcenverbrauch [100]- Hoher Speicherbedarf durch gebündelte Chromium Instanz; Sicherheitsrisiken durch JavaScript Bibliotheken; Performance bei 60k Pixel Bildern unterlegen.
Tkinter	Python Native	Veraltete UX [101]; Fehlende GPU-Beschleunigung für Canvas-Elemente; keine moderne HiDPI-Skalierung („blurry“ Text auf 4K-Monitoren).

**Tabelle 5.6:** Alternative zu PyQt5

Die Wahl von PyQt5 als Desktop-Framework adressiert die Anforderungen **FA-013** (GUI ohne CLI-Kenntnisse), **NFA-001** (On-Premise-Verarbeitung ohne Cloud) und **FA-012** (visuelle Validierung durch Overlay-System).

### 5.3.8 Datenpersistenz und Speicherschicht: PostgreSQL

Für die dauerhafte Speicherung der Analyseergebnisse und Gleisplandaten wurde das objekt-relationale Datenbanksystem (ORDBMS) PostgreSQL gewählt. Die Implementierung verfolgt einen hybriden Ansatz („Relational + Document Store“), der die Transaktionssicherheit einer SQL-Datenbank mit der Flexibilität einer dokumentenorientierten Speicherung (via JSONB) kombiniert.[102]

1. Schema-Design und Datenspeicherung: Das zentrale Speicherelement ist die Tabelle *workspaces*. Anstatt für jede erkannte Objektklasse (Signale, Weichen, Texte) separate relationale Tabellen anzulegen, werden die heterogenen Analyseergebnisse in einem semi-strukturierten Format aggregiert. Dieser Entwurf adressiert zwei spezifische Anforderungen der Gleisplananalyse:
  - Semi strukturierte Metadaten(JSONB): Die Ergebnisse der Objekterkennung (Bounding Boxes, Klassenwahrscheinlichkeiten, OCR-Texte) variieren stark in ihrer Struktur. Das JSONB-Format (Binary JSON) erlaubt es, diese hierarchischen Daten effizient zu speichern, ohne das Datenbankschema bei jeder Änderung der YOLO-Klassen anpassen zu müssen (Schema Evolution). Der GIN-Index (Generalized Inverted Index) ermöglicht dabei Abfragen auf tief verschachtelte Attribute (z. B. „Finde alle Pläne mit Signalen vom Typ AH“) in logarithmischer Zeit, vergleichbar mit klassischen Spaltenindizes.
  - Binäre Topologie-Daten (BYTEA): Das extrahierte Gleisskelett (der Graph der Fahr-

wege) liegt zur Laufzeit als speicheroptimiertes NumPy-Array vor. Die Konvertierung in Textform (JSON/CSV) wäre ineffizient und würde Präzision kosten. Daher wird dieses Array direkt als binäres Objekt (BYTEA)persistiert, was Ladezeiten minimiert und die exakte Repräsentation der Gleisgeometrie wahrt.

2. Versionierung und „Upsert“-Strategie: Um Dateninkonsistenzen bei mehrfacher Analyse desselben Plans zu vermeiden, implementiert die Anwendung eine UPSERT-Logik (Update or Insert)
3. Vorbereitung für Active Learning (Human in the Loop): Ein wesentliches Ziel der Architektur ist die nachhaltige Verbesserung der KI Modelle durch Nutzer Feedback (siehe Kapitel 8.3.2). Hierfür wurde ein separates Schema entworfen, das Abweichungen zwischen KI-Vorhersage und Nutzerkorrektur protokolliert:
4. Abgrenzung zu alternativen Speichertechnologien

Alternative	Evaluationsergebnis	Begründung der Ablehnung
SQLite	Bedingt geeignet	Obwohl leichtgewichtig, fehlt SQLite eine robuste Multi-User-Unterstützung (Database Locking bei Schreibzugriffen) und die JSON-Abfragefunktionen sind weniger performant als die GIN-Indizierung von PostgreSQL.[103]
MongoDB	Nicht gewählt	Als native NoSQL-DB wäre sie für JSON gut geeignet [104], bietet jedoch schwächere Garantien für relationale Integrität (z. B. Foreign Keys für user_corrections) und erhöht die Komplexität im Deployment (zusätzliche Infrastruktur-Komponente).
JSON-Files	Ungenügend	Die Speicherung in reinen Textdateien bietet keine Transaktionssicherheit (ACID), keine Indizierung für schnelle Suchen und führt bei parallelen Zugriffen zu „Race Conditions“.

**Tabelle 5.7:** Alternativen zu PostgreSQL

Das Datenbankdesign unterstützt die Anforderungen **NFA-005** (Prüfbarkeit durch Metadaten-Persistierung), **FA-011** (Änderungsverfolgung durch Versionierung) und **NFA-002** (Lizenzkonformität durch Open-Source-Datenbank).

## 5.4 Linking- & Assoziationsmodul

Die semantische Interpretation der Gleispläne erfordert die korrekte Verknüpfung zwischen erkannten Symbolen (Ankerobjekten) und den zugehörigen Texten (IDs, Koordinaten). Dieses Modul adressiert die Anforderungen **FA-006** (Fahrtrichtungsdetektion) und **FA-007** (Symbol-Koordinaten-Verknüpfung).

### 5.4.1 Konzeptioneller Ansatz

Das Linking-Modul basiert auf drei Kernprinzipien:

1. **Rotationsinvariante Geometrie:** Räumliche Beziehungen (z.B. „Text befindet sich unterhalb des Symbols“) werden im lokalen Koordinatensystem der OBB bewertet, nicht im globalen Bildkoordinatensystem. Dies gewährleistet robuste Verknüpfungen unabhängig von der Symbolorientierung.
2. **Klassenspezifische Regeln:** Für jede Objektklasse sind typische Text-Positionen definiert (z.B. Signalbezeichnungen oberhalb des Symbols, Koordinaten unterhalb). Diese Regeln basieren auf Domänenwissen über Gleisplan-Konventionen.
3. **Adaptive Mustererkennung:** Der Prototyp lernt wiederkehrende Layoutmuster aus erfolgreichen Verknüpfungen und nutzt diese zur Verbesserung zukünftiger Zuordnungen (vgl. Abschnitt 6.4.3).

### 5.4.2 Verknüpfungsstrategien

#### Proximity-basierte Suche

Textkandidaten werden innerhalb klassenspezifischer Suchradien um das Ankersymbol gesucht.

Die Auswahl erfolgt nach:

- Minimaler euklidischer Distanz
- Richtungskonformität (z.B. „unterhalb“ für Koordinaten)
- OCR-Konfidenz bei mehreren Kandidaten

#### Speziallogiken

Für komplexe Entitäten wurden spezialisierte Verknüpfungsalgorithmen entwickelt:

- **Fahrtrichtungserkennung:** Geometrische Ableitung aus der relativen Position von Signal und Gleiskoppelpule(festkodiert)
- **Haltepunkt-Gruppierung:** Kollinearitätstest zur Verknüpfung von Haltepunkt-Symbol, Signal und Koordinate

Die mathematische Formulierung und algorithmischen Details werden im Abschnitt 6.4 dargelegt.

## 5.5 Validierung und Qualitätssicherung

Das Validierungskonzept adressiert die Anforderungen **FA-008** (manuelle Korrektur durch Human-in-the-Loop), **NFA-004** (Robustheit durch Fehlerbehandlung) und **NFA-005** (Prüfbarkeit durch strukturierte Validierungsberichte). Das Validierungskonzept gliedert sich in drei Ebenen:

1. **Symbolvalidierung:** Prüfung erkannter Objekte gegen erwartete Bounding-Box-Geometrien (Größe, Seitenverhältnis, Kompaktheit).
2. **OCR-Validierung:** Regex-basierter Abgleich extrahierter Texte mit domänenspezifischen Mustern (Signal: [A-Z]{1,4}\d{1,4}, Koordinate: \d+[.,]\d+).
3. **Linking-Validierung:** Plausibilitätsprüfung räumlicher Assoziationen (Distanzschwellenwerte, Richtungskonformität).

Fehlgeschlagene Validierungen werden zur manuellen Prüfung markiert (Human-in-the-Loop).

Implementierung in Kapitel 6.5.

# 6. Implementierung

Dieses Kapitel beschreibt die technische Realisierung des Prototyps. Es werden die Implementierungsdetails der in Kapitel 5 konzipierten Architektur dargelegt, von der Datenaufbereitung über das Modelltraining bis zur finalen Datenaggregation.

## 6.1 Technologiestack und Entwicklungsumgebung

Die Implementierung des Prototyps basiert auf bewährten Open-Source-Technologien und -Bibliotheken. Die Auswahl des Technologiestacks erfolgte unter Berücksichtigung von Leistung, Wartbarkeit, Datenschutz und der Verfügbarkeit von Community-Support.

### 6.1.1 Programmiersprache und Kernbibliotheken

Die gesamte Implementierung erfolgte in **Python 3.9**, da diese Sprache ein umfangreiches Ökosystem für maschinelles Lernen, Bildverarbeitung und Datenmanipulation bietet und eine schnelle Prototypenentwicklung ermöglicht.

Komponente	Bibliothek	Version	Einsatzzweck
<i>Maschinelles Lernen</i>			
Deep Learning Objekterkennung	PyTorch Ultralytics	2.0.1 8.0.196	Framework für neuronale Netze YOLOv8 Implementierung
OCR (primär)	PaddleOCR	2.7.0	Hauptengine für Texterkennung
OCR (Fallback 1)	Tesseract	5.3.0	Fallback für niedrig aufgelöste oder verrauschte Texte
OCR (Fallback 2)	EasyOCR	1.7.0	Optionale dritte Engine
<i>Bildverarbeitung</i>			
Bildmanipulation	OpenCV	4.8.0	Preprocessing, Transformationen
Bildoperationen	Pillow	10.0.0	Zusätzliche Bildoperationen
PDF-Verarbeitung	PyMuPDF	1.23.3	PDF-zu-Bild Konvertierung
<i>Datenverarbeitung</i>			
Numerik	NumPy	1.24.3	Array-Operationen, Mathematik
Datenstrukturen	Pandas	2.0.3	Tabellarische Datenmanipulation
<i>Benutzeroberfläche &amp; Persistenz</i>			
GUI-Framework	PyQt5	5.15.9	Desktop-Anwendung
Datenbank	PostgreSQL	14.9	Datenpersistenz
ORM	psycopg2	2.9.7	PostgreSQL-Adapter

**Tabelle 6.1:** Übersicht der verwendeten Technologien und Bibliotheken

**Lizenzkonformität (NFA-002):** Alle verwendeten Bibliotheken unterliegen Open-Source-Lizenzen (Apache 2.0, MIT, BSD), die für den kommerziellen Einsatz bei Siemens Mobility geeignet sind. Eine detaillierte Lizenzprüfung wurde durchgeführt und bestätigt die Konformität mit den Unternehmensrichtlinien.

### 6.1.2 Entwicklungsumgebung

Die Entwicklung erfolgte in Visual Studio Code mit Git-Versionskontrolle. Die Annotation wurde mittels CVAT (Version 2.5.0) in lokaler Docker-Umgebung durchgeführt [105]. Abhängigkeiten sind in `requirements.txt` dokumentiert.

### 6.1.3 Hardware-Infrastruktur

Das YOLOv8-Training erfolgte auf einer GPU-Instanz (NVIDIA, AWS Instance g4dn.xlarge, 16 GB VRAM). Die finale Anwendung ist CPU-kompatibel (AMD Ryzen 5 PRO 5650U, 32 GB RAM), um eine breite Einsetzbarkeit ohne spezialisierte Hardware zu gewährleisten.

### 6.1.4 Projektstruktur

Die Implementierung folgt einer modularen Architektur mit klarer Trennung der Verantwortlichkeiten:

- `core/` - Kernlogik der Pipeline
  - `detection.py` - YOLO Objekterkennung
  - `ocr.py` - OCR-Engine Wrapper
  - `linking.py` - Symbol-Text Verknüpfung
- `ui/` - PyQt5 Benutzeroberfläche
- `utils/` - Hilfsfunktionen
- `validation/` - Validierungsregeln
- `database.py` - Datenbankzugriff
- `main.py` - Haupteinstiegspunkt

**Modularität (FA-014, NFA-008):** Die dargestellte Projektstruktur gewährleistet eine klare Trennung der Verantwortlichkeiten. Jedes Modul (`core/`, `ui/`, `validation/`) kann unabhängig weiterentwickelt werden. Die Integration neuer Symbolklassen erfordert lediglich Anpassungen in `core/detection.py` und `core/linking.py`, ohne die UI- oder Export-Logik zu modifizieren.

## 6.2 Objekterkennung mit YOLOv8-OBB

Die Objekterkennung bildet den ersten Schritt der Extraktionspipeline und ist für die zuverlässige Lokalisierung und Klassifikation aller relevanten Symbole in den Gleisplänen verantwortlich. Wie in den Grundlagen (Abschnitt 3.2.2) dargelegt, erfordert die rotationsvariierende Darstellung von Gleisplansymbolen den Einsatz von Oriented Bounding Boxes (OBB). Im Folgenden werden die Erstellung des Trainingsdatensatzes, die Durchführung und Optimierung des Modelltrainings sowie die daraus resultierenden Erkennungsleistungen beschrieben.

### 6.2.1 Datensatzerstellung und Annotation

Die Leistungsfähigkeit eines objekterkennenden Modells hängt unmittelbar von der Qualität und Quantität des Trainingsdatensatzes ab. Die Erstellung des Datensatzes erfolgte in einem mehrstufigen Prozess, der die Auswahl und Vorverarbeitung der Rohdaten, die präzise Annotation der Symbole, eine gezielte Datenaugmentation sowie die abschließende Aufteilung in Trainings- und Validierungsdaten umfasste. Insgesamt wurden **24 Gleispläne** von Siemens Mobility als Datengrundlage verwendet. Die Auswahl dieser Pläne erfolgte mit dem Ziel, eine möglichst breite Varianz hinsichtlich Streckenlänge, Symbolvielfalt und Planformatierung abzudecken.

#### Vorverarbeitung der Gleispläne

Die initialen Gleispläne lagen im PDF-Format vor und mussten zunächst in ein für die Annotation geeignetes Bildformat überführt werden. Hierzu wurde die Python-Bibliothek *PyMuPDF* eingesetzt, welche eine zuverlässige Rasterisierung von PDF-Dokumenten ermöglicht. Die Konvertierung erfolgte mit einer Auflösung von **500 DPI**, was einem Skalierungsfaktor von  $500/72 \approx 6,94$  gegenüber der Standard-PDF-Auflösung (72 DPI) entspricht. Diese hohe Auflösung wurde gewählt, um auch kleine Symbole wie Gleiskoppelpulpen oder Isolierstöße mit ausreichender Detailgenauigkeit darzustellen.

Nach der Rasterisierung wurden die hochauflösenden Gesamtbilder in kleinere Bildausschnitte (*Tiles*) segmentiert. Die gewählte Segmentgröße für die Annotation betrug  $2048 \times 2048$  Pixel, um ausreichend räumlichen Kontext für die präzise Markierung von Symbolen und deren Beziehungen zueinander zu gewährleisten. Für das anschließende Training und die Inferenz werden diese Tiles auf  $1024 \times 1024$  Pixel herunterskaliert, was der YOLO-Eingabegröße entspricht. Diese Strategie kombiniert die Vorteile großflächiger Annotation mit effizienter Modellverarbeitung: Die Annotationen auf  $2048 \times 2048$  Pixeln erfassen mehr Kontextinformation, während das Downsampling die Rechenzeit signifikant reduziert und den GPU-Speicherbedarf während des Trainings verringert, was größere Batch-Größen und damit stabilere Gradientenschätzungen ermöglicht. Bei der gewählten Renderauflösung von 500 DPI bleiben die relevanten visuellen Merkmale auch nach dem Downsampling differenzierbar. Die Dimensionierung stellt somit einen Kompromiss dar zwischen Annotationsqualität und effizienter Verarbeitung auf GPU-Systemen mit begrenztem Videospeicher (16 GB VRAM).

Aus den 24 Gleisplänen wurden nicht sämtliche Tiles entnommen, sondern eine gezielte Auswahl getroffen. Rein leere Hintergrund-Tiles sowie redundante Ausschnitte ohne relevante Symbole wurden aussortiert, um eine hohe Datensatzqualität sicherzustellen. Zudem wurde darauf geachtet, Tiles mit möglichst diversem Symbolbestand auszuwählen, um das Modell auf eine breite Palette realer Layoutkonfigurationen vorzubereiten. Durch diesen Selektionsprozess resultierten insgesamt **482 annotierte Bildausschnitte**, die die Grundlage des Datensatzes bilden.

## Annotationsprozess

Die Beschriftung erfolgte mittels CVAT [105] in einer lokalen Docker-Umgebung, um die Vertraulichkeit der Siemens-Mobility-Dokumente sicherzustellen. Jedes Symbol wurde mit einer Oriented Bounding Box (OBB) annotiert (vgl. Abschnitt 3.2.2), repräsentiert durch Zentrum ( $c_x, c_y$ ), Dimensionen ( $w, h$ ) und Rotationswinkel  $\theta$ :

$$\text{OBB} = (c_x, c_y, w, h, \theta) \quad (6.1)$$

CVAT exportiert im YOLO-OBB-Format, bei dem jede Bounding Box durch ihre vier normalisierten Eckpunktkoordinaten  $\{(x_i/W, y_i/H)\}_{i=1}^4$  beschrieben wird, wobei  $W$  und  $H$  die Bildbreite bzw. -höhe bezeichnen.

Der finale Datensatz umfasst insgesamt **13 Symbolklassen** gemäß Anforderung **FA-003**: *coordinate, signal, gks\_festkodiert, gks\_gesteuert, gm\_block, sverbinder, haltepunkt, weichen\_block, isolierstoß, prellbock, haltetafel, endeweichen* und *weichengruppeende*. Die Klasse *coordinate* nimmt eine besondere Rolle ein: Sie liefert die Positionsinformation (Kilometrierung), die den anderen 12 Klassen zugeordnet wird. Alle 13 Klassen werden in Kapitel 7 quantitativ evaluiert.

## Synthetische Datenaugmentation durch Rotation

Um die Robustheit des Modells gegenüber beliebigen Symbolorientierungen zu erhöhen und die Klassenbalance zu optimieren, wurde vor der Datensatzaufteilung eine gezielte synthetische Augmentation durch Rotation implementiert. Diese Strategie adressiert zwei zentrale Herausforderungen:

1. **Klassenungleichgewicht:** Die natürliche Verteilung der Symbolklassen in den 482 annotierten Original-Tiles ist stark unausgeglichen. Häufige Klassen wie *coordinate* dominieren, während seltene Klassen wie *endeweichen* oder *weichengruppeende* nur in wenigen Tiles vorkommen. Ohne Gegenmaßnahmen besteht die Gefahr, dass das Modell auf häufige Klassen overfittet und seltene Symbole unzureichend lernt.
2. **Orientierungsvielfalt:** Obwohl technische Zeichnungen Symbole in verschiedenen Winkeln enthalten, sind bestimmte Orientierungen – insbesondere  $0^\circ$  und  $90^\circ$  – in den realen Plänen überrepräsentiert, während diagonale Ausrichtungen seltener auftreten. Da das Modell in der Praxis Symbole unter beliebigen Rotationswinkeln zuverlässig erkennen muss, ist eine ausreichende Abdeckung aller relevanten Orientierungen im Trainingsdatensatz erforderlich.

**Augmentationsstrategie.** Die synthetische Rotation wurde selektiv auf Bildausschnitte angewendet, die Instanzen unterrepräsentierter Klassen enthalten. Dabei werden keine künstlichen Symbole erzeugt, sondern bestehende, real annotierte Tiles mitsamt ihren OBB-Annotationen

um definierte Winkel rotiert. Auf diese Weise trainiert das Modell dieselben Symbole in zusätzlichen Orientierungen. Für jede ausgewählte Tile wurden Rotationen um die folgenden Winkel durchgeführt:

$$\Theta_{\text{aug}} = \{-90^\circ, -60^\circ, -45^\circ, -30^\circ, -15^\circ, +15^\circ, +30^\circ, +45^\circ, +60^\circ, +90^\circ\} \quad (6.2)$$

Die Rotation erfolgte um den Bildmittelpunkt, wobei sowohl der Bildinhalt als auch sämtliche OBB-Annotationen mittels der Rotationsmatrix

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \quad (6.3)$$

entsprechend transformiert wurden. Jeder Eckpunkt  $(x_i, y_i)$  einer OBB wird relativ zum Rotationszentrum transformiert, sodass die geometrische Korrektheit der Annotation erhalten bleibt. In Kombination mit der OBB-Architektur wird durch die 10 Rotationswinkel eine vollständige Rotationsinvarianz von  $0^\circ$  bis  $360^\circ$  sichergestellt (vgl. Anforderung **FA-002**).

**Klassenspezifische Steuerung.** Die Anzahl der pro Tile generierten synthetischen Kopien wurde klassenabhängig gesteuert, um eine Zielverteilung von ca. 600 Instanzen pro unterrepräsentierter Klasse zu erreichen. Tiles, die ausschließlich bereits häufige Klassen wie *coordinate* enthalten, wurden nur moderat augmentiert, um deren ohnehin hohe Ausgangszahl nicht unverhältnismäßig weiter zu steigern. Bei stark unterrepräsentierten Klassen (z. B. *weichengruppeende* mit nur 74 Original-Instanzen oder *haltetafel* mit 93 Original-Instanzen) wurden bis zu 10 Rotationsvarianten pro Tile generiert.

**Technische Umsetzung.** Die Implementierung nutzt `cv2.getRotationMatrix2D()` und `cv2.warpAffine()` aus OpenCV. Die Bildgröße wurde dynamisch angepasst, um ein Abschneiden rotierter Objekte am Rand zu vermeiden. Die augmentierten Tiles wurden als separate Dateien mit eindeutiger Benennung (z. B. `*_a+30.jpg` für eine  $+30^\circ$ -Rotation) im Datensatz gespeichert, was eine deterministische Reproduzierbarkeit gewährleistet.

**Validierung der Augmentation.** Die transformierten OBB-Annotationen wurden stichprobenartig manuell auf geometrische Korrektheit überprüft, um sicherzustellen, dass die synthetische Rotation keine Artefakte einführt. Diese manuelle Validierung ist primär bei der *erstmaligen Implementierung* der Augmentationspipeline erforderlich, um die korrekte Funktionsweise der Rotationslogik und der OBB-Transformation zu verifizieren. Sobald die Pipeline einmal validiert ist, sind die geometrischen Transformationen deterministisch und mathematisch exakt, sodass bei der Hinzunahme neuer Symbolklassen oder der Anwendung auf weitere Gleisplantypen keine erneute manuelle Prüfung der Augmentation notwendig ist. Lediglich bei einer Änderung der Augmentationslogik selbst, etwa der Einführung zusätzlicher Transformationsarten wie

Skalierung oder Spiegelung, wäre eine erneute Validierung angezeigt.

**Ergebnis.** Durch die selektive Rotation wurden aus den 482 Original-Tiles zusätzlich **649 augmentierte Tiles** erzeugt, was den Datensatz auf insgesamt **1.131 Bildausschnitte** erweitert. Das Augmentationsverhältnis (57,4 % augmentiert) spiegelt den Fokus auf unterrepräsentierte Klassen wider: Tiles mit häufigen Symbolen wurden moderat oder gar nicht augmentiert, während Tiles mit seltenen Symbolen mehrfach rotiert wurden.

## Datenaufteilung und finale Statistiken

**Aufteilungsstrategie.** Die Aufteilung des augmentierten Gesamtdatensatzes erfolgte stratifiziert mit einem **80/20-Verhältnis** in Trainings- und Validierungsdaten. Die Stratifizierung stellt sicher, dass die relative Klassenverteilung in beiden Teilmengen annähernd gleich bleibt. Tabelle 6.2 zeigt die resultierende Aufteilung:

Teilmenge	Original	Augmentiert	Gesamt Bilder	Instanzen
Training	399	524	923	12.506
Validierung	83	125	208	3.076
<b>Gesamt</b>	<b>482</b>	<b>649</b>	<b>1.131</b>	<b>15.582</b>

**Tabelle 6.2:** Aufteilung des Datensatzes in Trainings- und Validierungssatz

Der Anteil augmentierter Bilder ist in beiden Teilmengen vergleichbar (56,8 % im Trainings- und 60,1 % im Validierungssatz), was die stratifizierte Aufteilung bestätigt.

**Klassenverteilung.** Tabelle 6.3 zeigt die vollständige Verteilung der Annotationen über alle 13 Symbolklassen, aufgeschlüsselt nach Trainings- und Validierungssatz. Die Spalte *Bilder* gibt an, in wie vielen Bildern der jeweiligen Teilmenge mindestens eine Instanz der betreffenden Klasse vorkommt. Da ein einzelnes Bild typischerweise mehrere Symbolklassen gleichzeitig enthält, überlappen sich die Einzelwerte der Spalte *Bilder* und summieren sich nicht zur Gesamtbildzahl.

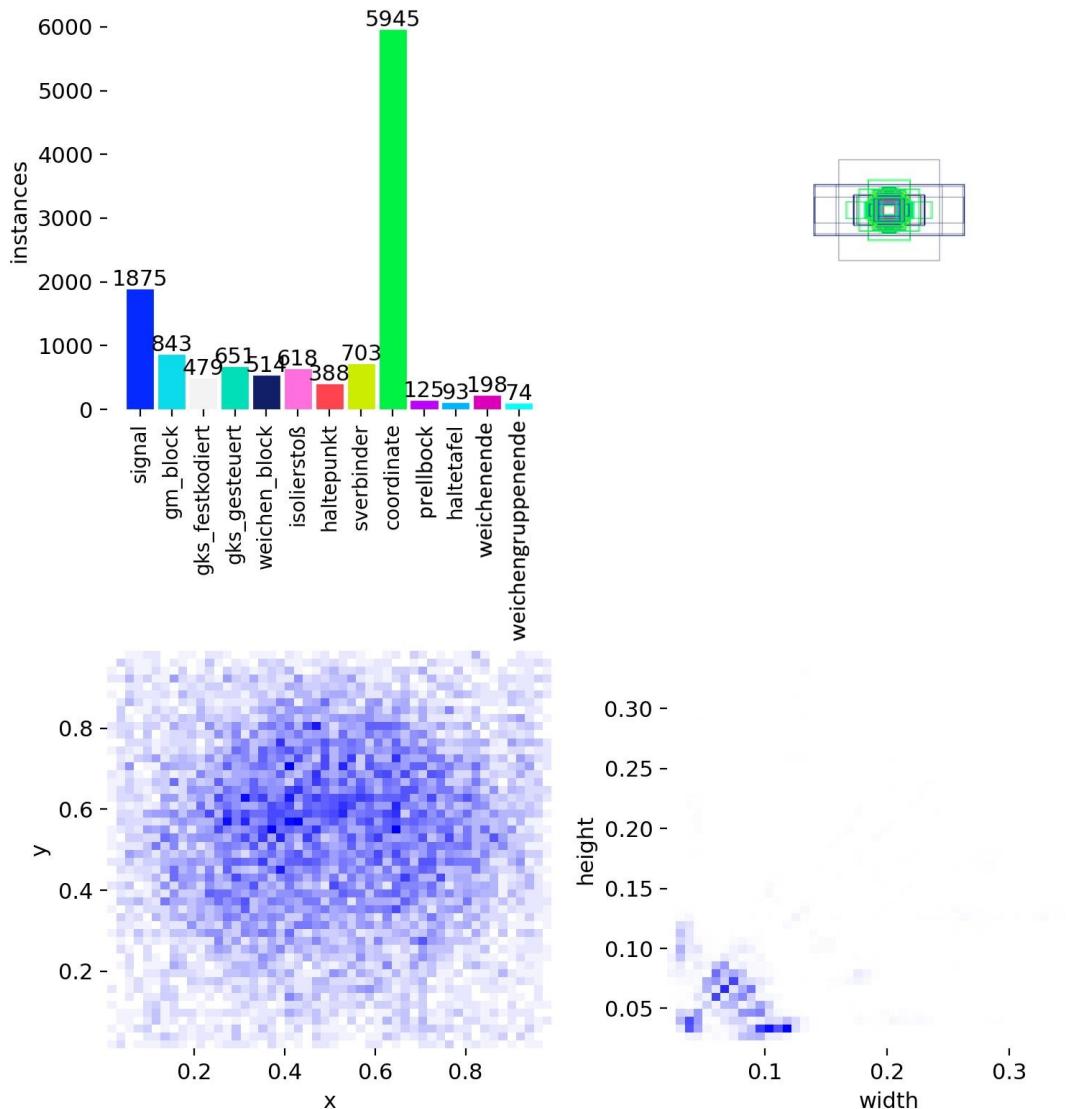
Klasse	Train Bild.	Train Inst.	Val Bild.	Val Inst.	Gesamt
coordinate	890	5.945	207	1.483	7.428
signal	681	1.875	164	473	2.348
gm_block	585	843	143	201	1.044
gks_gesteuert	586	651	147	157	808
gks_festkodiert	390	479	88	122	601
sverbinder	611	703	140	150	853
isolierstoß	264	618	67	147	765
weichen_block	325	514	67	129	643
haltepunkt	376	388	92	92	480
endeweichen	170	198	50	50	248
prellbock	109	125	31	31	156
haltetafel	81	93	23	23	116
weichengruppeende	74	74	18	18	92
<b>Gesamt</b>	<b>923*</b>	<b>12.506</b>	<b>208*</b>	<b>3.076</b>	<b>15.582</b>

**Tabelle 6.3:** Verteilung der Annotationen über alle 13 Klassen im Trainings- und Validierungssatz. Die Spalte *Bilder* zeigt die Anzahl der Bilder mit mindestens einer Instanz der jeweiligen Klasse; da Bilder mehrere Klassen gleichzeitig enthalten, ist die Summe der Einzelwerte größer als die mit \* markierte Gesamtbildzahl.

Die Daten wurden in eine standardisierte COCO-Verzeichnisstruktur (Abbildung 6.1) überführt, welche die Organisation von Trainings- und Validierungsdaten sowie die zugehörigen Annotationen in separaten Verzeichnissen vorsieht.



**Abbildung 6.1:** COCO-konforme Datensetstruktur für das YOLOv8-OBB-Training



**Abbildung 6.2:** Charakteristika des Trainingsdatensatzes (923 Bilder, 12.506 Instanzen): (oben links) Klassenverteilung mit absoluten Instanzzahlen, (unten links) räumliche Verteilung der Bounding-Box-Zentren, (unten rechts) Größenverteilung der annotierten Objekte in normalisierten Koordinaten.

Abbildung 6.2 visualisiert die Charakteristika des Trainingsdatensatzes. Die Klassenverteilung zeigt die nach Augmentation resultierende Verteilung mit der erwartungsgemäß dominierenden Koordinaten-Klasse (5.945 Trainingsinstanzen), gefolgt von Signalen (1.875 Trainingsinstanzen). Die räumliche Verteilung der Bounding-Box-Zentren zeigt eine gleichmäßige Abdeckung des Bildbereichs ohne systematische Häufungen oder Leerräume, was auf eine gute Repräsentativität des Datensatzes hindeutet. Die Größenverteilung verdeutlicht die charakteristisch kleinen Objektdimensionen der Gleisplansymbole, die überwiegend im Bereich von 2–10 % der normalisierten Bilddimensionen liegen.

**Diskussion der Klassenverteilung.** Die **Dominanz der Koordinaten-Klasse** ist ein inhärentes Merkmal von Gleisplänen und kein Artefakt des Datensatzes: Grundsätzlich besitzt jedes Objekt in einem Gleisplan – ob Signal, GKS, Weiche oder Haltepunkt – eine eigene Kilometerangabe. Die Anzahl der Koordinatenbeschriftungen entspricht daher annähernd der Summe aller übrigen Objektinstanzen. Ein direkter Größenvergleich zwischen der Koordinaten-Klasse und einzelnen Symbolklassen ist somit methodisch nur bedingt aussagekräftig, da Koordinaten funktional eine Referenzklasse darstellen, die alle anderen Symbolklassen begleitet.

Die **Instanzzahlen einzelner Klassen** variieren entsprechend ihrer natürlichen Häufigkeit in Gleisplänen. Aus eisenbahntechnischer Sicht wäre zu erwarten, dass S-Verbinder und Isolierstoße in realen Gleisplänen mindestens ebenso häufig vorkommen wie Signale, da jedes Signal theoretisch einen S-Verbinder oder Isolierstoß in unmittelbarer Nähe besitzt, ergänzt durch weitere Vorkommen am Ende des Durchrutschweges, in Weichenbereichen und als Trennung mehrerer Gleisabschnitte [5]. Die im Datensatz beobachteten Instanzzahlen spiegeln die tatsächliche Verteilung in den annotierten 24 Gleisplänen wider. Die detaillierte quantitative Evaluation der Erkennungsleistung über alle 13 Klassen erfolgt in Kapitel 7.

### 6.2.2 Modelltraining und Optimierung

#### Modellarchitektur und Hyperparameter

Als Modellarchitektur wurde **YOLOv8I-OBB** gewählt, die *Large*-Variante der YOLOv8-Familie mit OBB-Erweiterung. Diese Variante bietet mit ca. 43,7 Millionen Parametern eine hohe Modellkapazität, die für die zuverlässige Unterscheidung der 13 visuell teilweise ähnlichen Symbolklassen – insbesondere der GKS-Varianten – erforderlich ist. Im Vergleich zur kleineren *Small*-Variante (11,2 Mio. Parameter) erreicht die *Large*-Variante eine höhere Erkennungsgenauigkeit bei feingranularen Klassenunterscheidungen, wobei die Inferenzgeschwindigkeit für den Batch-Verarbeitungsmodus der Gleisplanpipeline weiterhin ausreichend ist.

Das Training wurde auf einer cloud-basierten GPU-Instanz (NVIDIA, 16 GB Videospeicher) durchgeführt, da lokale Entwicklungsmaschinen nicht über ausreichende Rechenleistung verfügen, um das Training in angemessener Zeit abzuschließen. Diese Cloud-Ressource ist jedoch *ausschließlich für die initiale Trainingsphase* erforderlich. Das resultierende trainierte Modell (best.pt, ca. 87 MB) kann anschließend auf Standard-Hardware ohne GPU für die Inferenz eingesetzt werden (vgl. Abschnitt 6.2.3). Für zukünftige Projekte ergeben sich zwei Szenarien: Wenn ein bestehendes Modell bereits die benötigten Symbolklassen abdeckt, ist *kein erneutes Training* notwendig – das Modell wird direkt wiederverwendet. Nur bei der Erweiterung um neue, bisher nicht trainierte Symbolklassen oder bei der Anpassung an einen grundlegend anderen Gleisplantyp ist ein erneutes Training auf einer GPU-Instanz erforderlich. Durch den Einsatz von Transfer Learning (vgl. Tabelle 6.4) kann dabei auf den bereits trainierten Gewichten aufgebaut werden, was die benötigte Trainingszeit und Datenmenge für solche Erweiterungen deutlich reduziert.

Das Modell wurde mit auf dem COCO-Datensatz [12] vortrainierten Gewichten initialisiert (*Transfer Learning*), um die in den frühen Netzwerkschichten bereits gelernten generischen Merkmalsextraktoren – wie Kanten-, Ecken- und Texturerkennung – zu nutzen und die Konvergenz auf dem domänenspezifischen Gleisplan-Datensatz zu beschleunigen.

Tabelle 6.4 fasst die verwendeten Hyperparameter zusammen:

Parameter	Wert	Begründung
Modell	YOLOv8l-OBB	Hohe Kapazität für 13 Klassen
Vortraining	COCO-Weights	Transfer Learning
Eingabegröße	1024 × 1024 px	Angepasst an Tile-Dimensionen
Epochen	120	Konvergenz ab Epoche ~60
Batch-Größe	8	Limitiert durch 16 GB VRAM
Optimizer	SGD	YOLOv8-Standard
Initiale Lernrate	0,01	YOLOv8-Default
Finale Lernrate	0,0002	Cosine Annealing (lrf = 0,01)
Warmup-Epochen	3	Stabilisierung der frühen Gradienten
Momentum	0,937	Standard-Konfiguration
Weight Decay	0,0005	Regularisierung

**Tabelle 6.4:** Hyperparameter des YOLOv8l-OBB-Trainings

Die **Eingabegröße** von 1024 × 1024 Pixel weicht vom YOLOv8-Standard (640 Pixel) ab und wurde bewusst gewählt, um der Kleinteiligkeit der Gleisplansymbole gerecht zu werden. Bei einer Standard-Eingabegröße von 640 Pixel würden die bereits kleinen Symbole (vgl. Größenverteilung in Abbildung 6.2) weiter verkleinert und könnten unterhalb der effektiven Erkennungsgrenze des Feature-Extractors fallen.

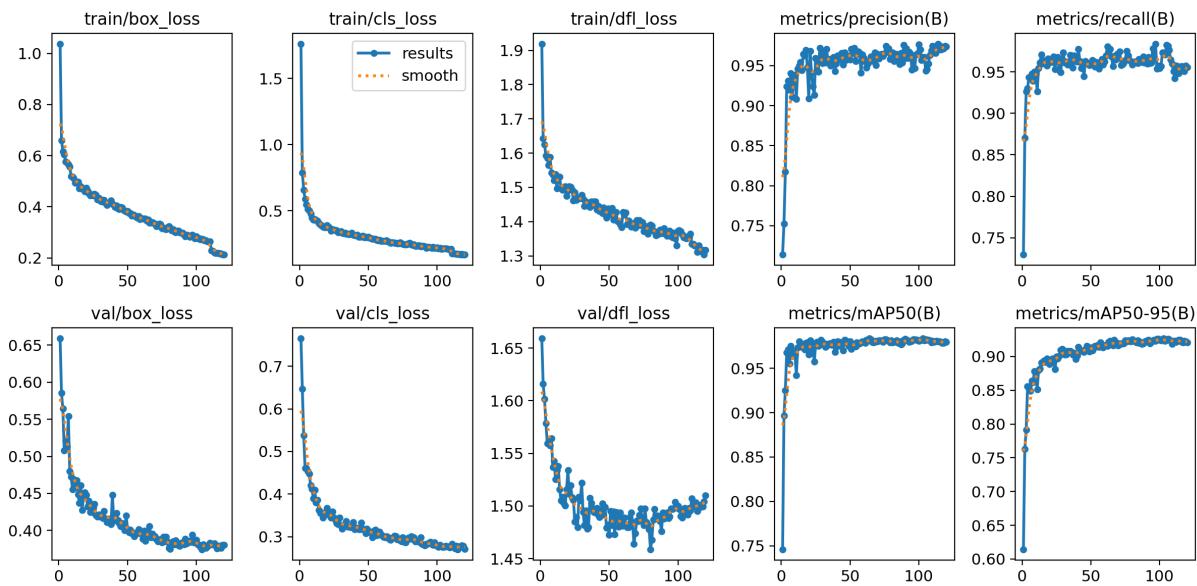
Die **Batch-Größe** von 8 wurde durch den verfügbaren GPU-Speicher (16 GB VRAM) begrenzt. Die *Large*-Variante des YOLOv8-OBB-Modells belegt mit 43,7 Millionen Parametern bereits einen erheblichen Anteil des GPU-Speichers. In Kombination mit der Eingabegröße von 1024 × 1024 Pixeln, den zugehörigen Aktivierungen und Gradienten ist eine Batch-Größe von 8 das Maximum innerhalb des verfügbaren Speicherbudgets.

Das **Lernraten-Scheduling** folgt dem in YOLOv8 implementierten *Cosine Annealing* mit linearer Warmup. In den ersten drei Epochen wird die Lernrate linear von einem niedrigen Wert auf die initiale Lernrate von 0,01 erhöht, um instabile Gradienten in der Frühphase des Trainings zu vermeiden. Anschließend fällt die Lernrate gemäß einer Kosinus-Funktion kontinuierlich auf den finalen Wert von 0,0002 (1 % der Initialrate) ab. Dieses Schema ermöglicht in der mittleren Trainingsphase eine effiziente Exploration des Parameterraums, während in der Spätphase eine feinere Justierung der Gewichte durch die reduzierte Schrittweite erfolgt.

## Trainingskonvergenz

Die Gesamtdauer des Trainings über 120 Epochen betrug **ca. 6 Stunden**. YOLOv8-OBB optimiert drei Verlustkomponenten simultan:

- **Box Loss** (*train/box\_loss*): Quantifiziert die Abweichung zwischen vorhergesagten und tatsächlichen Bounding-Box-Positionen und -Dimensionen.
- **Classification Loss** (*train/cls\_loss*): Misst den Fehler bei der Klassenzuordnung der erkannten Objekte.
- **Distribution Focal Loss** (*train/dfl\_loss*): Optimiert die Vorhersageverteilung der Bounding-Box-Eckpunkte und verbessert insbesondere die Lokalisierungsgenauigkeit bei kleinen Objekten.



**Abbildung 6.3:** Trainings- und Validierungsverläufe über 120 Epochen: (oben) Trainings-Losses und Metriken, (unten) Validierungs-Losses und mAP-Werte.

Abbildung 6.3 zeigt den Verlauf aller drei Verlustkomponenten sowie der Evaluationsmetriken über die 120 Trainingsepochen. Der Trainingsverlauf lässt sich in drei charakteristische Phasen unterteilen:

1. **Warmup-Phase (Epochen 1–3):** Die Lernrate wird linear auf den Initialwert hochgefahren. Alle Verlustkomponenten fallen steil ab, was die schnelle Adaption der vortrainierten COCO-Gewichte an die Gleisplan-Domäne zeigt. Bereits nach Epoche 3 erreicht das Modell eine mAP@0.5 von 92,5 %.
2. **Primäre Lernphase (Epochen 4–60):** Die Verluste sinken stetig, während Precision, Recall und mAP kontinuierlich ansteigen. Die mAP@0.5 steigt von 92,5 % auf ca. 98,0 %, die mAP@0.5:0.95 von 79,1 % auf ca. 92,0 %. In dieser Phase lernt das Modell die domänenspezifischen Merkmale der 13 Symbolklassen.
3. **Feintuning-Phase (Epochen 61–120):** Die Validierungs metriken stabilisieren sich auf hohem Niveau mit geringfügigen Schwankungen. Die Trainingsverluste sinken weiter – insbesondere zeigt sich bei Epoche ~110 ein markanter Abfall (*box\_loss*: 0,26→0,21;

*cls\_loss*: 0,21 → 0,17), der auf die finale Phase des Cosine-Annealing-Schedules zurückzuführen ist, in der die stark reduzierte Lernrate eine feinere Gewichtsoptimierung ermöglicht.

**Overfitting-Analyse.** Ein entscheidendes Qualitätskriterium des Trainings ist die Abwesenheit von Overfitting, d. h. das Modell generalisiert auf ungewohnte Daten (Validierungssatz) ähnlich gut wie auf Trainingsdaten. Die Validierungsverluste zeigen ab Epoche 50 einen stabilen Plateauverlauf ohne Anstieg: *val/box\_loss* stabilisiert sich bei ca. 0,38, *val/cls\_loss* bei ca. 0,28 und *val/dfl\_loss* bei ca. 1,50. Da ein ansteigender Validierungsverlust bei gleichzeitig sinkendem Trainingsverlust das klassische Indiz für Overfitting wäre, kann dieses Phänomen für das vorliegende Training ausgeschlossen werden. Der moderate und stabile Gap zwischen Trainings- und Validierungsverlust (z. B. *box\_loss*: 0,21 vs. 0,38) ist für augmentierte Datensätze typisch und reflektiert die zusätzliche Variabilität der Trainingsdaten durch die synthetische Rotation.

## Trainingsergebnisse

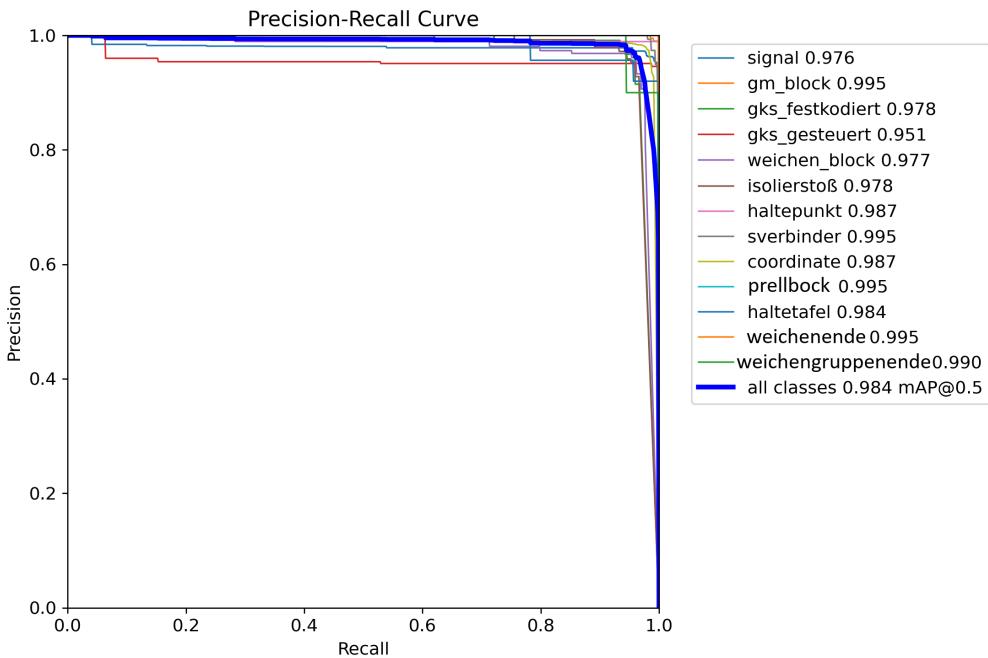
**Aggregierte Metriken.** Tabelle 6.5 fasst die finalen Detektionsmetriken des trainierten Modells zusammen, evaluiert auf dem Validierungsdatensatz (208 Bilder, 3.076 Instanzen):

Metrik	Wert
Precision (Durchschnitt, alle Klassen)	97,4 %
Recall (Durchschnitt, alle Klassen)	95,6 %
F1-Score (bei Konfidenzschwelle 0,674)	96,0 %
mAP@0.5 (alle Klassen)	98,4 %
mAP@0.5:0.95 (alle Klassen)	92,7 %

**Tabelle 6.5:** Aggregierte Detektionsmetriken auf dem Validierungsdatensatz (208 Bilder)

Die berichteten Metriken beziehen sich auf das automatisch gesicherte Modell mit den besten Validierungsergebnissen (Epoche 103 von 120), welches als `best.pt` exportiert und für die produktive Inferenz verwendet wird. Die mAP@0.5 von **98,4 %** zeigt, dass das Modell bei einem IoU-Schwellenwert von 50 % nahezu alle Symbole korrekt lokalisiert und klassifiziert. Die strengere Metrik mAP@0.5:0.95, die über IoU-Schwellenwerte von 50 % bis 95 % mittelt, erreicht **92,7 %** und bestätigt eine hohe Lokalisierungspräzision auch bei strengen Überlappungsanforderungen. Der optimale F1-Score von 0,96 wird bei einer Konfidenzschwelle von 0,674 erreicht, was darauf hindeutet, dass das Modell überwiegend hochkonfidente Vorhersagen trifft.

**Klassenspezifische Leistung.** Abbildung 6.4 zeigt die Precision-Recall-Kurven für alle 13 Symbolklassen. Die klassenspezifischen AP@0.5-Werte sind in Tabelle 6.6 zusammengestellt.



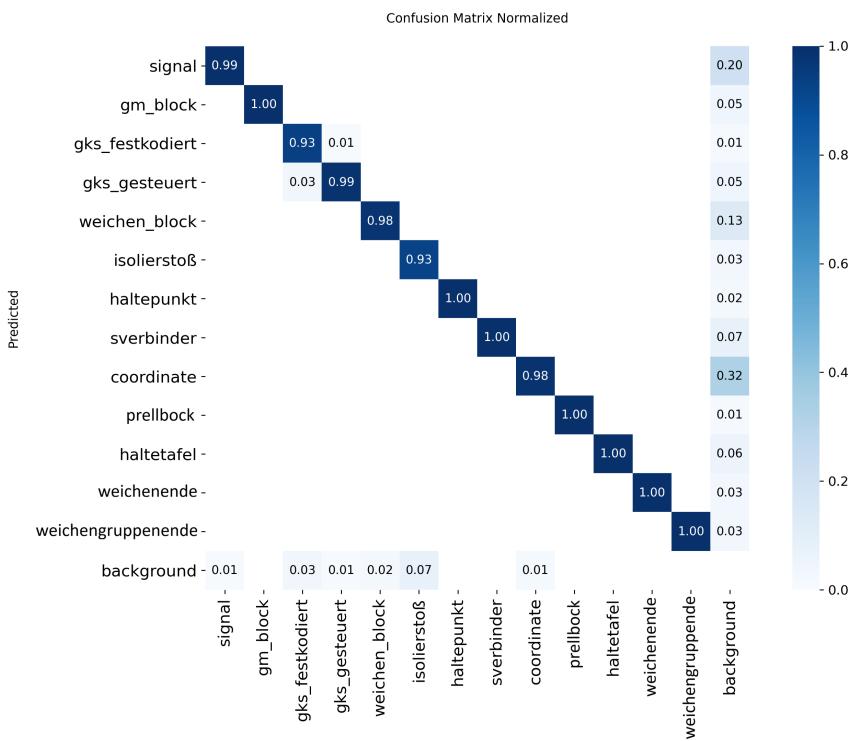
**Abbildung 6.4:** Precision-Recall-Kurven für alle 13 Symbolklassen auf dem Validierungsdatensatz. Die Werte in der Legende geben die jeweilige AP@0.5 an.

Klasse	Val Inst.	AP@0.5	Bewertung
gm_block	201	99,5 %	Exzellent
sverbinder	150	99,5 %	Exzellent
prellbock	31	99,5 %	Exzellent
endeweichen	50	99,5 %	Exzellent
weichengruppeende	18	99,0 %	Exzellent
coordinate	1.483	98,7 %	Exzellent
haltepunkt	92	98,7 %	Exzellent
haltetafel	23	98,4 %	Exzellent
gks_festkodiert	122	97,8 %	Sehr gut
isolierstoß	147	97,8 %	Sehr gut
weichen_block	129	97,7 %	Sehr gut
signal	473	97,6 %	Sehr gut
gks_gesteuert	157	95,1 %	Gut
∅ Alle Klassen	<b>3.076</b>	<b>98,4 %</b>	

**Tabelle 6.6:** Klassenspezifische AP@0.5 auf dem Validierungsdatensatz, sortiert nach absteigender AP@0.5

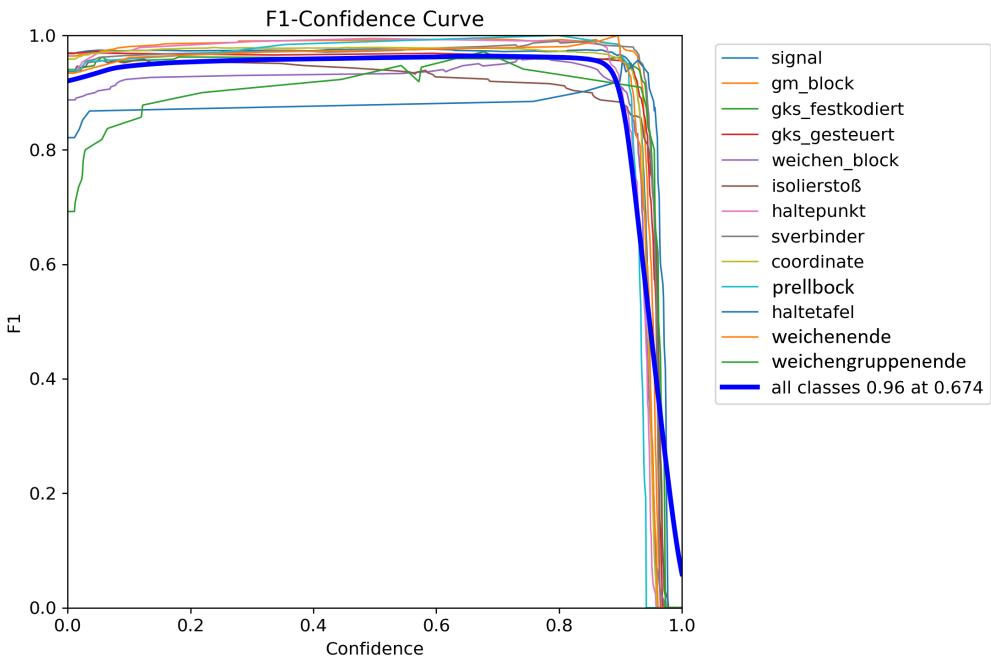
Bemerkenswert ist, dass **alle 13 Klassen eine AP@0.5 über 95 % erreichen** und die Streuung mit 95,1 % bis 99,5 % sehr gering ist. Dies bestätigt, dass die in Abschnitt 6.2.1 beschriebene Augmentationsstrategie das Klassenungleichgewicht erfolgreich adressiert hat: Auch Klassen mit wenigen Validierungsinstanzen (z. B. *weichengruppeende* mit nur 18 Instanzen) erreichen eine AP@0.5 von 99,0 %.

**Fehleranalyse.** Abbildung 6.5 zeigt die normalisierte Konfusionsmatrix auf dem Validierungsdatensatz. Die Diagonale zeigt durchgehend hohe Werte ( $\geq 0,93$ ), was eine zuverlässige Klassifikation über alle Klassen bestätigt. Die vereinzelten Fehlklassifikationen betreffen primär die visuell ähnlichen GKS-Varianten sowie kleine Symbole wie Isolierstöße an Tile-Grenzen. Eine detaillierte klassenspezifische Analyse der Detektionsleistung erfolgt in Kapitel 7.



**Abbildung 6.5:** Normalisierte Konfusionsmatrix auf dem Validierungsdatensatz. Zeilen repräsentieren die vorhergesagten Klassen, Spalten die tatsächlichen Klassen.

**Konfidenzkalibrierung.** Abbildung 6.6 zeigt die F1-Score-Konfidenz-Kurve für alle 13 Klassen.



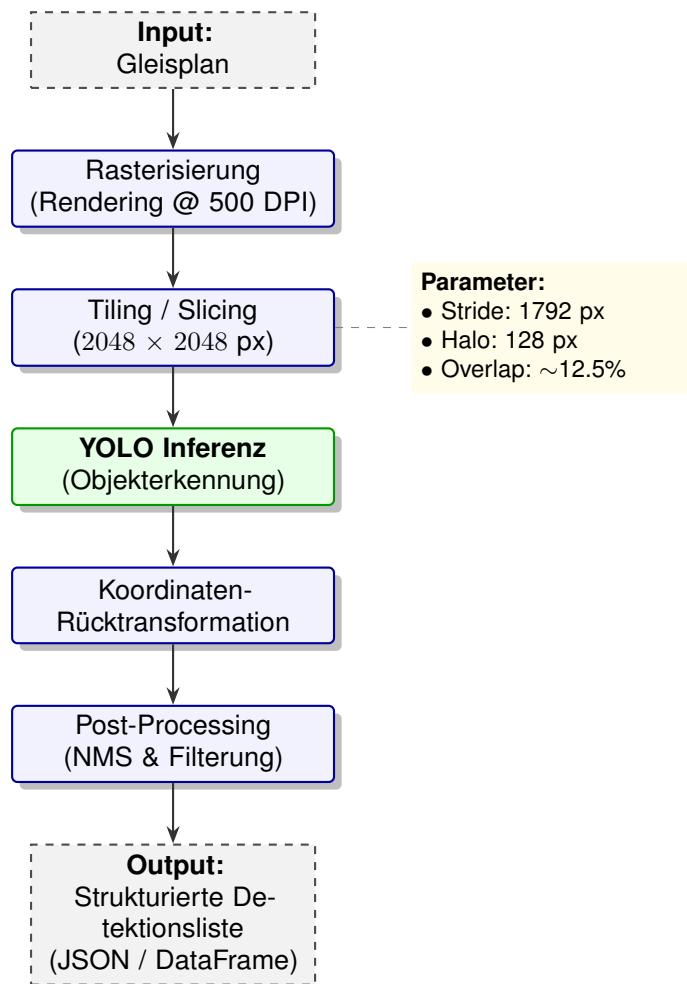
**Abbildung 6.6:** F1-Score in Abhängigkeit der Konfidenzschwelle. Das globale Optimum liegt bei  $F1 = 0,96$  bei einer Schwelle von 0,674.

Die Kurve zeigt, dass die Mehrzahl der Klassen über einen breiten Konfidenzbereich (0,1–0,9) einen F1-Score über 0,95 aufrechterhält, was auf eine gute Kalibrierung des Modells hindeutet. Der optimale globale F1-Score von 0,96 bei einer Konfidenzschwelle von 0,674 wird als Standard-Detektionsschwelle für die nachfolgende Inferenz-Pipeline übernommen. Eine detaillierte Evaluation der End-to-End-Leistung einschließlich der nachgelagerten OCR- und Verknüpfungsschritte erfolgt in Kapitel 7.

Das trainierte Modell (`best.pt`) wurde als Gewichtsdatei exportiert und in die Inferenz-Pipeline des Prototyps integriert. Die im folgenden Abschnitt beschriebene Inferenz-Pipeline nutzt dieses Modell zur Verarbeitung neuer, bisher ungesehener Gleispläne.

### 6.2.3 Inferenz-Pipeline Implementierung

Nach Abschluss des Trainings wurde das Modell mit den optimalen Gewichten (`best.pt`) in die produktive Extraktionspipeline integriert. Die Inferenz auf neuen, zuvor ungesehenen Gleisplänen erfolgt in mehreren aufeinander aufbauenden Schritten. Abbildung 6.7 zeigt den vollständigen Ablauf der Inferenz-Pipeline von der PDF-Eingabe über das Tiling bis zur Rücktransformation der globalen Koordinaten.



**Abbildung 6.7:** Ablauf der Inferenz-Pipeline

### Eingabeverarbeitung und Formatnormalisierung

Der erste Schritt der Inferenz-Pipeline besteht in der Überführung der Eingabedaten in ein einheitliches Rasterformat. Das System unterscheidet dabei zwischen zwei Verarbeitungspfaden entsprechend dem Eingabeformat (Anforderung **NFA-009**).

#### PDF-Verarbeitung:

Für PDF-Dateien wird die Bibliothek PyMuPDF (fitz) verwendet. Für jede Seite wird eine Transformationsmatrix definiert, die die Skalierung von der Standard-PDF-Auflösung (72 DPI) auf die Zielauflösung von 500 DPI beschreibt:

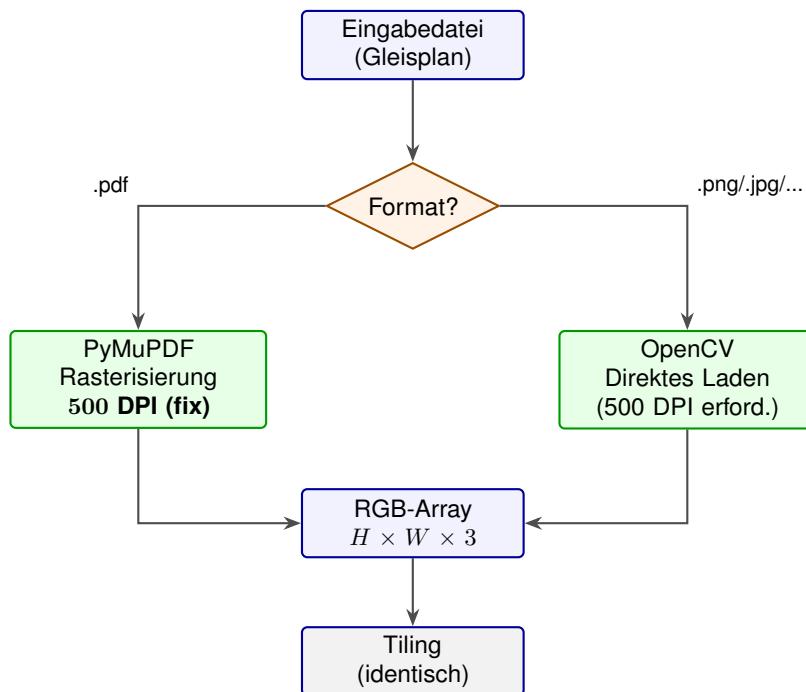
$$\text{Skalierungsfaktor} = \frac{500}{72} \approx 6,94 \quad (6.4)$$

Die resultierendePixmap wird in ein RGB-Array überführt. Ein typischer A0-Gleisplan resultiert bei 500 DPI in einem Bild von ca.  $30000 \times 5000$  bis  $90000 \times 7000$  Pixeln.

#### Bildverarbeitung:

Für Rastergrafiken (PNG, JPEG, TIFF, BMP) werden die nativen Pixeldaten mittels OpenCV direkt geladen. Da das YOLOv8-OBB Modell ausschließlich auf 500-DPI-Daten trainiert wurde, ist für das aktuelle Gleisplan-Layout eine Eingabeauflösung von 500 DPI zwingend erforderlich. Bei abweichenden Auflösungen sind erhebliche Erkennungsfehler zu erwarten. Die Unterstützung von Bilddateien dient primär der zukünftigen Erweiterbarkeit für andere Layouts mit potenziell anderen Auflösungsanforderungen.

Abbildung 6.8 zeigt den formatabhängigen Verarbeitungspfad vor der einheitlichen Tiling-Pipeline.



**Abbildung 6.8:** Formatabhängige Eingabeverarbeitung vor der einheitlichen Tiling-Pipeline.

### Einheitliche Weiterverarbeitung:

Nach der formatspezifischen Eingabeverarbeitung liegt das Bild als RGB-Array vor. Alle nachfolgenden Schritte – Tiling, YOLO-Inferenz, Koordinaten-Rücktransformation und NMS – sind unabhängig vom ursprünglichen Eingabeformat identisch.

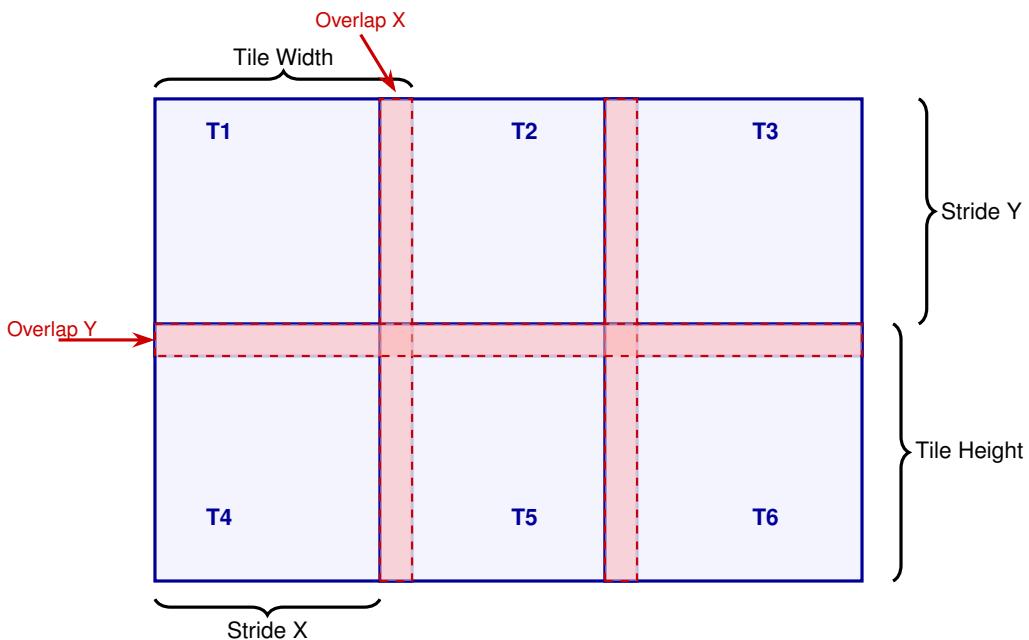
### Tiling-Strategie

Da die resultierenden Bilder mit mehreren Millionen Pixeln die Eingabegröße des YOLO-Modells ( $1024 \times 1024$ ) bei weitem überschreiten, wird das Vollbild in überlappende Kacheln (Tiles) zerlegt. Die Tile-Größe beträgt  $T = 2048 \times 2048$  Pixel, was eine Verarbeitung auf gängigen GPUs ermöglicht und gleichzeitig ausreichend Kontext für die Detektion bietet.

**Downsampling:** Die extrahierten Tiles mit einer Größe von  $2048 \times 2048$  Pixeln werden vor der YOLO-Inferenz auf die Trainingsgröße von  $1024 \times 1024$  Pixeln herunterskaliert. Downsampling (auch Unterabtastung oder Dezimation genannt) bezeichnet in der digitalen Bildverarbeitung die

Reduktion der räumlichen Auflösung eines Bildes durch Verringerung der Pixelanzahl [53]. Bei einem Skalierungsfaktor von 2 wird jedes  $2 \times 2$  Pixelfeld des Originalbildes zu einem einzelnen Pixel im Zielbild zusammengefasst, wobei typischerweise Interpolationsverfahren wie bilineare oder bikubische Interpolation zum Einsatz kommen, um Aliasing-Artefakte zu minimieren.

Die resultierenden Bounding-Box-Koordinaten werden anschließend mit dem Faktor 2 multipliziert, um die korrekten Positionen im Originaltile zu erhalten. Dieses Vorgehen ermöglicht eine höhere effektive Auflösung bei der Symbolerkennung, da feinere Details im größeren Eingabebild erhalten bleiben, bevor das Downsampling erfolgt.



**Abbildung 6.9:** 2D-Tiling-Strategie mit quadratischen Kacheln

Die Überlappung zwischen benachbarten Tiles beträgt  $O = 12.5\%$ , was einer absoluten Überlappung von 256 Pixeln bei einer Tile-Größe von 2048 Pixeln entspricht. Der Stride (Schrittweite) zwischen Tiles ergibt sich zu:

$$S = T \cdot (1 - O) = 2048 \cdot 0.875 = 1792 \text{ Pixel} \quad (6.5)$$

Abbildung 6.9 illustriert diese Strategie, bei der durch die identische Schrittweite in X- und Y-Richtung eine gleichmäßige Abdeckung mit definiertem Überlappungsbereich gewährleistet wird. Die Überlappung ist essenziell, um zu verhindern, dass Symbole, die nahe an Tile-Grenzen liegen, durch die Segmentierung geteilt werden und somit nicht oder nur unvollständig detektiert werden können. Ein Symbol, das im Überlappungsbereich liegt, wird von mindestens zwei benachbarten Tiles erfasst und hat somit eine hohe Chance, in mindestens einem der Tiles vollständig sichtbar zu sein.

Die Berechnung der Tile-Positionen erfolgt durch Iteration über die Koordinaten:

$$\begin{aligned} X &= \{x \mid x \in \{0, S, 2S, \dots\} \wedge x + T \leq W\} \cup \{W - T\} \\ Y &= \{y \mid y \in \{0, S, 2S, \dots\} \wedge y + T \leq H\} \cup \{H - T\} \end{aligned} \quad (6.6)$$

Die Ergänzung der Mengen um  $\{W - T\}$  bzw.  $\{H - T\}$  stellt sicher, dass auch der rechte und untere Bildrand vollständig abgedeckt werden, selbst wenn sie nicht durch reguläre Schritte erreichbar sind.

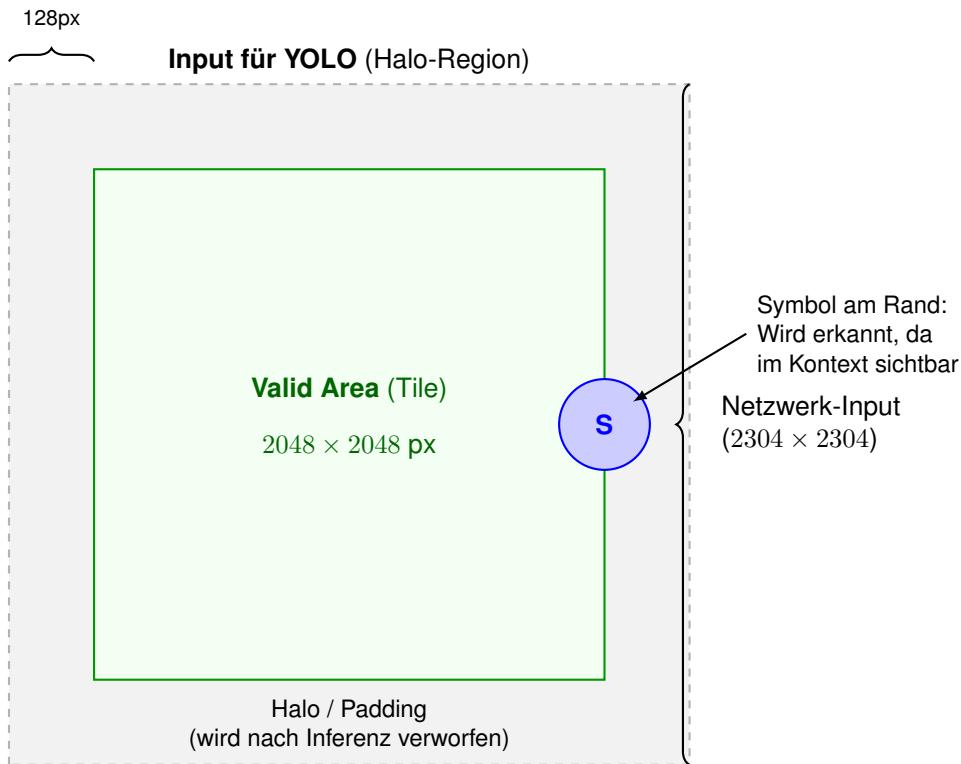
Zusätzlich zur Überlappung wird ein sogenannter Halo um jedes Tile herum hinzugefügt. Der sogenannte Halo bezeichnet einen zusätzlichen Randbereich von  $H_{\text{halo}} = 128$  Pixeln, der bei der Bildextraktion mitgeladen wird:

$$\text{Tile}_{\text{mit Halo}} = \text{Bild}[y - H_{\text{halo}} : y + T + H_{\text{halo}}, x - H_{\text{halo}} : x + T + H_{\text{halo}}] \quad (6.7)$$

Der Halo stellt zusätzlichen Kontext für Symbole am Tile-Rand bereit, wird jedoch bei der späteren Filterung der Detektionen nicht berücksichtigt. Nur Detektionen, deren Zentrum innerhalb der eigentlichen Tile-Grenzen (ohne Halo) liegt, werden akzeptiert:

$$\text{Akzeptiere Detektion} \Leftrightarrow x \leq c_x \leq x + T \wedge y \leq c_y \leq y + T \quad (6.8)$$

wobei  $(c_x, c_y)$  das Zentrum der detektierten Bounding Box bezeichnet. Abbildung 6.10 veranschaulicht dieses Prinzip: Das neuronale Netz (YOLO) erhält einen erweiterten Bildausschnitt (grau dargestellt), um angeschnittene Objekte am Rand korrekt zu erkennen. Gewertet werden jedoch nur Detektionen, deren Zentrum im inneren Bereich (grün dargestellt) liegt.



**Abbildung 6.10:** Visualisierung der Halo-Strategie

Der Halo-Bereich von 128 Pixeln dient als zusätzlicher Kontextbereich um jedes Tile und ist unabhängig vom Overlap. Während der Overlap (256 Pixel) die Mindestüberlappung zwischen benachbarten Tiles definiert, um Objekte an Tile-Grenzen zu erfassen, stellt der Halo sicher, dass Objekte nahe dem Tile-Rand ausreichend Kontextinformation für eine zuverlässige Erkennung haben. Detektionen, deren Zentrum im Halo-Bereich liegt, werden beim NMS-Schritt gefiltert.

### Winkeladaptive Parameterauswahl

Eine Besonderheit der Implementierung besteht in der winkeladaptiven Anpassung der Bounding-Box-Parameter für jede Detektion. Nach der initialen YOLO-Detektion, die eine OBB in Form von  $(c_x, c_y, w, h, \theta)$  liefert, werden die Dimensionen  $w$  und  $h$  sowie das Padding basierend auf dem Rotationswinkel  $\theta$  angepasst.

Zunächst wird der Winkel normalisiert, um eine kanonische Repräsentation zu erhalten. Die Normalisierung erfolgt durch:

$$\theta_{\text{norm}} = \begin{cases} \theta - \frac{\pi}{2}, & \text{falls } \theta > \frac{\pi}{4} \\ \theta + \frac{\pi}{2}, & \text{falls } \theta < -\frac{\pi}{4} \\ \theta, & \text{sonst} \end{cases} \quad (6.9)$$

Diese Normalisierung stellt sicher, dass der Winkel im Bereich  $[-\frac{\pi}{4}, \frac{\pi}{4}]$  liegt. Falls eine Rotation außerhalb dieses Bereichs erforderlich ist, werden Breite und Höhe vertauscht:

$$(w_{\text{norm}}, h_{\text{norm}}) = \begin{cases} (h, w), & \text{falls Rotation durchgeführt} \\ (w, h), & \text{sonst} \end{cases} \quad (6.10)$$

Basierend auf dem normalisierten Winkel  $\theta_{\text{norm}}$  wird entschieden, ob das Symbol als horizontal, vertikal oder angular orientiert klassifiziert wird:

$$\text{Orientierung} = \begin{cases} \text{Horizontal,} & \text{falls } |\theta_{\text{norm}}| < 15^\circ \text{ oder } |\theta_{\text{norm}} - 180^\circ| < 15^\circ \\ \text{Vertikal,} & \text{falls } |\theta_{\text{norm}} - 90^\circ| < 15^\circ \text{ oder } |\theta_{\text{norm}} - 270^\circ| < 15^\circ \\ \text{Angular,} & \text{sonst} \end{cases} \quad (6.11)$$

Für jede Orientierung werden unterschiedliche Expansionsfaktoren  $(e_x, e_y)$  und Padding-Werte  $p$  angewendet:

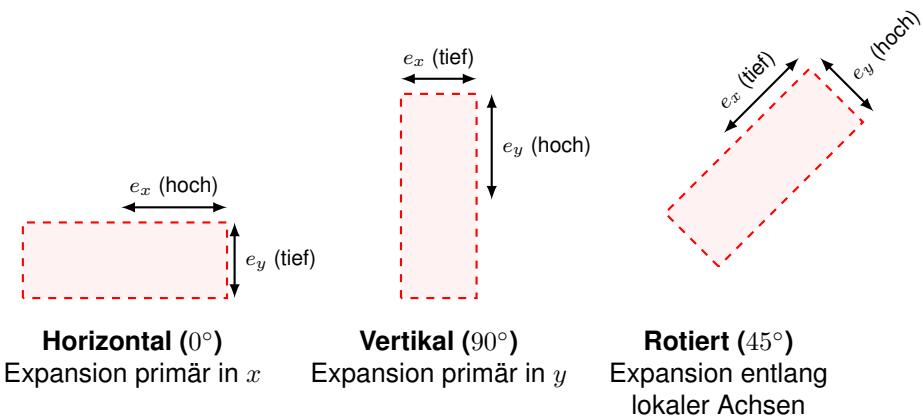
Orientierung	$e_x$	$e_y$	$p$ [Pixel]
Horizontal	2.5	1.5	16
Vertikal	1.5	2.5	16
Angular	2.0	2.0	24

**Tabelle 6.7:** Winkeladaptive Bounding-Box-Parameter

Die effektiven Dimensionen der Bounding Box werden dann berechnet als:

$$\begin{aligned} w_{\text{eff}} &= w_{\text{norm}} \cdot e_x + 2p \\ h_{\text{eff}} &= h_{\text{norm}} \cdot e_y + 2p \end{aligned} \quad (6.12)$$

Diese Anpassung ist kritisch für die nachfolgende OCR, da bei horizontal orientierten Texten mehr horizontaler Kontext benötigt wird (z.B. für Texte wie „Signal A123“), während bei vertikal orientierten Texten mehr vertikaler Kontext erforderlich ist. Angular orientierte Objekte erhalten symmetrische Expansion und erhöhtes Padding, um Rotationseffekte auszugleichen.



**Abbildung 6.11:** Adaptive Expansion der Region of Interest (ROI)

Abbildung 6.11 visualisiert diese Anpassung für die drei Orientierungskategorien. Im Fall angular orientierter Objekte (rechts) erfolgt die Expansion entlang der rotierten Objektachsen.

### Post-Processing und Duplikatentfernung

Nach der Inferenz auf allen Tiles müssen die Detektionen aggregiert und Duplikate entfernt werden. Die Koordinaten-Rücktransformation erfolgt durch Addition der Tile-Offsets:

$$\begin{aligned} c_x^{\text{global}} &= c_x^{\text{lokal}} + x_{\text{Tile}} \\ c_y^{\text{global}} &= c_y^{\text{lokal}} + y_{\text{Tile}} \end{aligned} \quad (6.13)$$

Aufgrund der Überlappung zwischen Tiles werden Objekte im Überlappungsbereich mehrfach detektiert. Zur Eliminierung dieser Duplikate wird eine Non-Maximum Suppression (NMS) durchgeführt. Die NMS arbeitet klassenweise und behält für jede Klasse nur die Detektionen mit höchster Konfidenz bei, wenn mehrere Detektionen dasselbe Objekt beschreiben.

Der Überlappungsgrad zweier Bounding Boxes wird mittels Intersection over Union (IoU) quantifiziert:

$$\text{IoU}(B_1, B_2) = \frac{\text{Area}(B_1 \cap B_2)}{\text{Area}(B_1 \cup B_2)} \quad (6.14)$$

Der NMS-Algorithmus funktioniert wie folgt:

1. Sortiere alle Detektionen einer Klasse nach absteigender Konfidenz
2. Wähle die Detektion mit höchster Konfidenz und füge sie zur Ergebnismenge hinzu
3. Entferne alle Detektionen, deren IoU mit der gewählten Detektion einen klassenspezifischen Schwellenwert  $\tau_{\text{NMS}}$  überschreitet
4. Wiederhole Schritte 2-3, bis keine Detektionen mehr vorhanden sind

Die NMS-Schwellenwerte sind klassenspezifisch konfiguriert:

Klasse	$\tau_{\text{NMS}}$	Begründung
coordinate	0.3	Strenger Schwellenwert, da Koordinaten oft dicht gruppiert sind
signal	0.5	Standard-Schwellenwert für mittlere Objektdichte
weichen_block	0.6	Lockerer Schwellenwert für große Symbole mit natürlicher Varianz
default	0.5	Fallback für nicht explizit konfigurierte Klassen

**Tabelle 6.8:** Klassenspezifische NMS-Schwellenwerte

Zusätzlich zur NMS werden Detektionen unterhalb klassenspezifischer Konfidenzschwellen verworfen. Diese Schwellen wurden empirisch auf dem Validierungssatz optimiert, um ein Gleichgewicht zwischen Precision und Recall zu erreichen.

### Ausgabeformat

Das Ergebnis der Objekterkennungsstufe ist eine strukturierte Liste aller detektierten Symbole, wobei jede Detektion durch folgende Attribute charakterisiert ist:

- **Klassenname** und **Klassenindex**: Semantische Kategorie des Symbols
- **Konfidenz**: Wahrscheinlichkeit der Korrektheit der Detektion im Intervall  $[0, 1]$
- **AABB-Koordinaten**  $(x_1, y_1, x_2, y_2)$ : Achsenparallele Bounding Box für UI-Darstellung
- **OBB-Parameter**  $(c_x, c_y, w_{\text{eff}}, h_{\text{eff}}, \theta_{\text{norm}})$ : Rotierte Bounding Box für OCR
- **Polygon-Eckpunkte**  $\{(x_i, y_i)\}_{i=1}^4$ : Exakte Umrisspolygone
- **Seitennummer**: Zuordnung zur entsprechenden PDF-Seite

Diese strukturierte Repräsentation dient als Eingabe für die nachfolgende OCR-Stufe, welche die Textinformationen in unmittelbarer Nähe der detektierten Symbole extrahiert.

### Performance-Charakteristika

Die durchschnittliche Inferenzzeit pro Tile beträgt ca. 120 ms auf einer Standard-CPU (AMD Ryzen 5 PRO 5650U, 6 Kerne, 32 GB RAM). Für einen typischen A0-Gleisplan mit ca. 40-50 Tiles ergibt sich eine Gesamtverarbeitungszeit von ca. 5-6 Sekunden für die reine YOLO-Inferenz; die vollständige Pipeline inklusive OCR und Linking benötigt durchschnittlich 12.3 Minuten (vgl. Kapitel 7). Diese CPU-basierte Inferenz wurde bewusst gewählt, um die Einsetzbarkeit der Software auf Standard-Workstations ohne dedizierte GPU zu gewährleisten, was die Verbreitung und Nutzung im Produktivumfeld erleichtert.

## 6.3 Orientierungsadaptive OCR-Pipeline

Die Texterkennung aus den detektierten Symbolen stellt eine zentrale Herausforderung dar, da die Textregionen beliebig im Gleisplan rotiert sein können. Dieses Kapitel beschreibt die implementierte OCR-Pipeline, die durch Multi-Engine-Kaskadierung, rotationsadaptive Strategien und klassenspezifische Bildvorverarbeitung eine robuste Texterkennung ermöglicht und damit die Anforderungen **FA-004** (OCR-Genauigkeit) und **FA-005** (OCR-Robustheit) adressiert.

### 6.3.1 Multi-Engine Kaskadierung

Das System implementiert eine kaskadierende Architektur mit drei OCR-Engines: PaddleOCR als primäre Engine, Tesseract als Fallback-Engine und optional EasyOCR für schwierige Fälle. Die Grundidee dieser Kaskadierung ist, dass verschiedene OCR-Engines unterschiedliche Stärken haben. PaddleOCR ist schnell und präzise bei gut ausgerichteten Texten, Tesseract ist besonders robust bei verrauschten oder pixeligen Bildern, und EasyOCR ist spezialisiert auf schwierige Winkel und perspektivische Verzerrungen.

#### Funktionsweise der Kaskadierung

Die Kaskadierung funktioniert nach einem sequenziellen Prinzip. Zunächst wird PaddleOCR auf dem Bildausschnitt ausgeführt, und zwar viermal mit unterschiedlichen Rotationswinkeln:  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  und  $270^\circ$ . Der Grund für diese vier Rotationen ist, dass PaddleOCR am besten mit horizontal ausgerichtetem Text funktioniert. Durch das Ausprobieren aller vier Hauptrichtungen wird sichergestellt, dass mindestens eine Rotation den Text horizontal ausrichtet.

Für jede dieser Rotationen wird das OCR-Ergebnis durch eine klassenspezifische Validierungsfunktion geprüft. Diese Validierung überprüft, ob der erkannte Text dem erwarteten Muster der jeweiligen Symbolklasse entspricht. Beispielsweise muss ein Signal-Text mit einem Großbuchstaben beginnen, gefolgt von 1-3 Ziffern (wie „A12“ oder „B234“). Zusätzlich zur Mustererkennung wird der Konfidenz-Wert (Vertrauenswert) des OCR-Ergebnisses überprüft.

Der Konfidenz-Wert von PaddleOCR liegt typischerweise zwischen 0 und etwa 2, wobei höhere Werte eine größere Sicherheit bedeuten. Ein empirisch bestimmter Schwellenwert von 1.3 hat sich als optimal erwiesen: Ergebnisse über diesem Wert sind in der Regel korrekt, während niedrigere Werte oft Fehler enthalten. Dieser Schwellenwert wurde durch Tests auf dem Trainingsdatensatz bestimmt und stellt einen guten Kompromiss zwischen Präzision (wie viele der als „korrekt“ markierten Ergebnisse wirklich korrekt sind) und Recall (wie viele der tatsächlich vorhandenen Texte erkannt werden) dar.

Wenn PaddleOCR bei keiner der vier Rotationen ein validiertes Ergebnis mit ausreichender Konfidenz liefert, wird Tesseract als Fallback-Engine aktiviert. Tesseract verwendet einen anderen Algorithmus als PaddleOCR und ist besonders robust gegenüber verrauschten oder pixeligen Bildern. Tesseract wird mit dem Page Segmentation Mode 7 konfiguriert, der das Bild

als einzelne Textzeile behandelt-eine Annahme, die für die kurzen Symbol-Beschriftungen im Gleisplan zutreffend ist.

Falls auch Tesseract kein valides Ergebnis liefert und die optionale dritte Engine EasyOCR aktiviert ist, wird diese als letzte Instanz aufgerufen. EasyOCR ist besonders gut bei stark geneigten oder perspektivisch verzerrten Texten, benötigt aber erheblich mehr Rechenzeit (etwa 5-10 Mal langsamer als PaddleOCR).

### Konfiguration der OCR-Engines

Jede OCR-Engine wurde mit spezifischen Parametern konfiguriert, um optimale Ergebnisse für die Domäne der Gleispläne zu erzielen.

**PaddleOCR** wurde mit den folgenden Einstellungen verwendet:

- **Detection Model:** „ch\_PP-OCRv4\_det“ - ein vortrainiertes Modell zur Textlokalisierung, das ursprünglich für chinesische Dokumente entwickelt wurde, aber auch für deutsche/englische Texte ausgezeichnet funktioniert
- **Recognition Model:** „ch\_PP-OCRv4\_rec“ - das zugehörige Erkennungsmodell für die eigentliche Zeichenerkennung
- **Angle Classification:** Aktiviert - ermöglicht automatische Erkennung und Korrektur der Textorientierung
- **Sprache:** „en“ (Englisch) - da die meisten Beschriftungen alphanumerisch sind

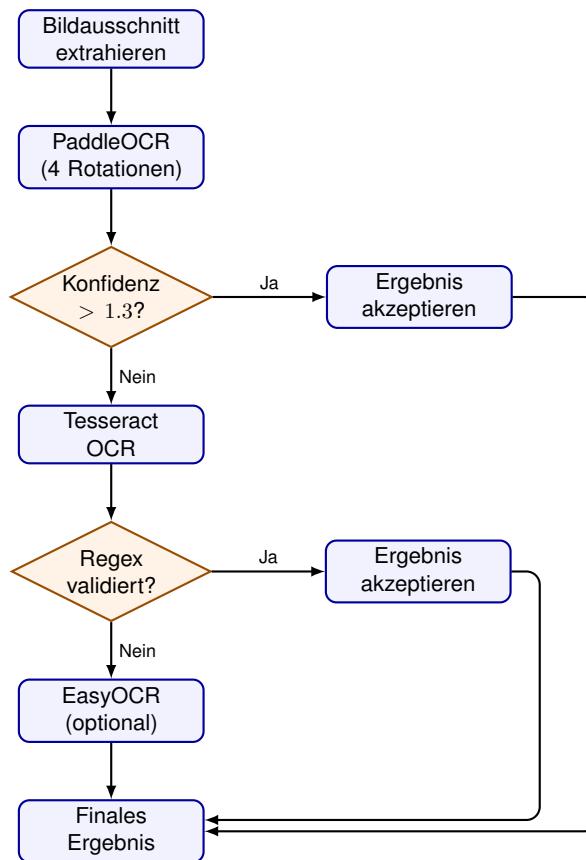
**Tesseract** wurde mit den folgenden Parametern konfiguriert:

- **Page Segmentation Mode (psm):** 7 - behandelt das Bild als einzelne Textzeile, was für kurze Symbol-Beschriftungen optimal ist
- **OCR Engine Mode (oem):** 3 - verwendet den LSTM-basierten neuronalen Netzwerk-Modus, der die beste Genauigkeit bietet
- **Character Whitelist:** Beschränkt auf „0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ+-.“ - dies verhindert, dass Tesseract falsche Sonderzeichen oder Kleinbuchstaben erkennt, die in den meisten Symbolklassen nicht vorkommen

Die Whitelist ist besonders wichtig, da sie die Fehlerrate deutlich reduziert. Ohne diese Einschränkung würde Tesseract manchmal exotische Sonderzeichen oder Umlaute erkennen, die in technischen Gleisplänen nie vorkommen.

**EasyOCR** wurde mit minimaler Konfiguration verwendet:

- **Sprachen:** [„en“] - Englisch als Basissprache
- **GPU-Nutzung:** Aktiviert - beschleunigt die Verarbeitung erheblich, wenn eine GPU verfügbar ist



**Abbildung 6.12:** Ablaufdiagramm der Multi-Engine OCR-Kaskadierung

### 6.3.2 Dual-Winkel-Routing System

Eine zentrale Herausforderung bei der OCR ist die beliebige Rotation der Symbole im Gleisplan. Während einige Symbole horizontal ausgerichtet sind, können andere um  $45^\circ$ ,  $30^\circ$  oder sogar beliebige Winkel gedreht sein. Um diese Vielfalt zu bewältigen, wurde ein Dual-Path-Routing-System implementiert, das zwischen zwei verschiedenen Verarbeitungswegen unterscheidet.

#### Cardinal Path und Angular Path

Das System unterscheidet zwischen dem **Cardinal Path** (für nahezu horizontal/vertikal ausgerichtete Texte) und dem **Angular Path** (für beliebig gedrehte Texte). Die Entscheidung, welcher Pfad verwendet wird, basiert auf dem Rotationswinkel der Bounding Box.

Zunächst wird der von YOLO gelieferte Winkel normalisiert. Da YOLO Winkel im Bereich von  $0^\circ$  bis  $360^\circ$  liefern kann, werden Winkel über  $180^\circ$  durch Subtraktion von  $360^\circ$  in den Bereich  $-180^\circ$  bis  $180^\circ$  transformiert. Ein Winkel von  $270^\circ$  wird beispielsweise zu  $-90^\circ$ . Dies vereinfacht die Logik erheblich, da nun positive Winkel eine Drehung gegen den Uhrzeigersinn und negative Winkel eine Drehung im Uhrzeigersinn darstellen.

Anschließend wird geprüft, ob der Absolutwert des normalisierten Winkels unter  $15^\circ$  liegt. Diese  $15^\circ$  wurden empirisch durch Experimente bestimmt: Bei Winkeln bis zu dieser Größe liefert

eine einfache Rotation auf den nächsten kardinalen Winkel ( $0^\circ, 90^\circ, 180^\circ$  oder  $270^\circ$ ) bessere Ergebnisse als eine komplexe Perspektiventransformation. Der Grund ist, dass bei kleinen Winkeln die Perspektiventransformation zu leichten Verzerrungen führen kann, während die diskrete Rotation verlustfrei ist.

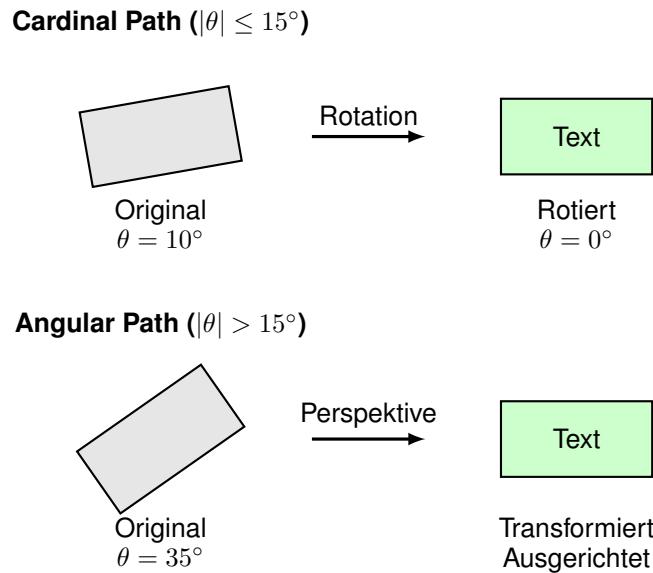
**Cardinal Path (Winkel  $\leq 15^\circ$ ):** Die vier kardinalen Kandidaten ( $0^\circ, 90^\circ, 180^\circ, 270^\circ$ ) sind jeweils  $90^\circ$  voneinander entfernt. Der Schwellenwert von  $15^\circ$  bedeutet, dass der gemessene Winkel maximal  $15^\circ$  vom nächstgelegenen Kardinalwinkel abweicht. Da die Kardinalen  $90^\circ$  auseinander liegen, deckt dieser Schwellenwert einen Bereich von  $\pm 15^\circ$  um jeden Kardinalwinkel ab – also insgesamt  $30^\circ$  von den  $90^\circ$  zwischen zwei benachbarten Kardinalen. Bei Winkeln innerhalb dieses Bereichs wird der Bildausschnitt einfach auf den nächstgelegenen kardinalen Winkel rotiert. Die Rotation erfolgt mittels einer Standard-Rotationsmatrix, die das Bild um den Mittelpunkt dreht. Diese Methode ist schnell, einfach und vermeidet Interpolationsartefakte, da nur  $90^\circ$ -Schritte verwendet werden.

Konkret wird aus den vier Kandidaten derjenige gewählt, der die kleinste Winkeldifferenz zum normalisierten Winkel hat. Ein Winkel von  $12^\circ$  liegt innerhalb des  $15^\circ$ -Schwellenwerts und wird daher auf  $0^\circ$  rotiert; ein Winkel von  $85^\circ$  wird auf  $90^\circ$  rotiert, da die Differenz nur  $5^\circ$  beträgt.

**Angular Path (Winkel  $> 15^\circ$ ):** Bei größeren Abweichungen von den kardinalen Winkeln wird eine Perspektiventransformation verwendet. Diese Methode nutzt die vier Eckpunkte der Oriented Bounding Box (OBB), um eine Transformationsmatrix zu berechnen. Die Transformation wandelt die gedrehte OBB in ein achsenparalleles Rechteck um, wodurch der Text horizontal ausgerichtet wird.

Die Perspektiventransformation wird mit OpenCV's `getPerspectiveTransform()`-Funktion implementiert. Diese Funktion benötigt zwei Sätze von je vier Punkten: die Quellpunkte (die vier Ecken der gedrehten OBB) und die Zielpunkte (die vier Ecken eines aufrechten Rechtecks). Aus diesen acht Punkten berechnet die Funktion eine  $3 \times 3$ -Transformationsmatrix, die dann mit `warpPerspective()` auf das Bild angewendet wird.

Obwohl diese Methode rechenintensiver ist (etwa 2-3 Mal langsamer als eine einfache Rotation), liefert sie bei beliebigen Winkeln deutlich bessere Ergebnisse, da der Text exakt horizontal ausgerichtet wird, nicht nur annähernd.



**Abbildung 6.13:** Vergleich: Cardinal Path vs. Angular Path Rotationsstrategien

Abbildung 6.14 demonstriert die erfolgreiche Verarbeitung eines um  $37,5^\circ$  rotierten Koordinaten-texts durch den Angular Path.

Das System erkannte den Koordinatentext  $0.0629$  korrekt trotz der nicht-kardinalen Orientierung von  $\theta = -37,5^\circ$ . Die Perspektiventransformation richtete die gedrehte Oriented Bounding Box horizontal aus, wodurch PaddleOCR mit einer Konfidenz von 0.95 den Text extrahieren konnte. Die erfolgreiche Verknüpfung mit dem benachbarten Symbol (Gl.112) validiert sowohl die OCR-Robustheit (FA-005) als auch die Rotationsinvarianz (FA-002) für beliebige Winkel außerhalb der kardinalen Orientierungen.



**Abbildung 6.14:** Verarbeitung eines um  $37,5^\circ$  rotierten Elemente durch Angular-Path-OCR

### 6.3.3 Klassenspezifische Strategien

Verschiedene Symbolklassen im Gleisplan haben unterschiedliche visuelle Eigenschaften, die spezielle Vorverarbeitungsschritte erfordern. Signale haben beispielsweise oft einen farbigen Hintergrund und größere Beschriftungen, während GKS häufig von dünnen Linien umgeben sind und sehr kompakte Texte haben. Um diese Unterschiede zu berücksichtigen, wurde für jede Klasse eine spezialisierte Verarbeitungsstrategie implementiert.

#### Adaptive Padding

Das Padding erweitert den von YOLO gelieferten Bildausschnitt, um sicherzustellen, dass der gesamte Text erfasst wird. Oft liegt der Text teilweise außerhalb der detektierten Bounding Box, besonders bei Signalen, wo die Beschriftung über oder neben dem eigentlichen Symbol sein kann.

Die Größe des Paddings wird primär relativ zur Größe der Bounding Box berechnet. Um jedoch auch bei sehr schmalen Objekten (z.B. Signalmasten) eine Mindest erfassung zu gewährleisten, wird zusätzlich ein absolutes Minimum  $p_{\min}=10$  Pixel definiert. Die Berechnung erfolgt durch:

$$p_x = \max(\alpha \cdot w_{\text{bbox}}, p_{\min}), \quad p_y = \max(\beta \cdot h_{\text{bbox}}, p_{\min}) \quad (6.15)$$

wobei  $p_x$  das horizontale Padding (in Pixeln),  $p_y$  das vertikale Padding (in Pixeln),  $w_{\text{bbox}}$  die Breite der Bounding Box,  $h_{\text{bbox}}$  die Höhe der Bounding Box, und  $\alpha$  sowie  $\beta$  klassenspezifische Faktoren sind.

Die folgende Tabelle zeigt die optimierten Faktoren für jede Symbolklasse:

Klasse	$\alpha$	$\beta$	Begründung
Signal	0.2	0.3	Beschriftung oft oberhalb des Symbols
GKS (beide Typen)	0.15	0.15	Kompakte, zentrierte Beschriftung
Weiche Koordinate	0.1 0.25	0.1 0.25	Text direkt am Symbol Längere Zahlenfolgen, mehr Platz nötig

**Tabelle 6.9:** Klassenspezifische Padding-Faktoren und deren Begründung

Für Signale wurde beispielsweise ein Faktor von  $\alpha = 0.2$  (20% der Breite) und  $\beta = 0.3$  (30% der Höhe) als optimal ermittelt. Das größere vertikale Padding berücksichtigt, dass Signalbeschriftungen typischerweise oberhalb des Symbols positioniert sind. Bei einer Bounding Box von  $100 \times 50$  Pixeln würde dies ein Padding von 20 Pixeln horizontal und 15 Pixeln vertikal bedeuten.

Das finale erweiterte Bild hat dann die Dimensionen  $(h + 2p_y) \times (w + 2p_x)$ , wobei das „2“

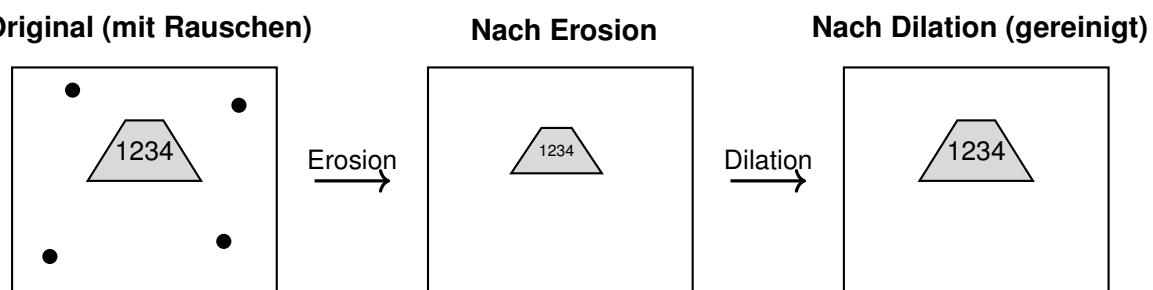
berücksichtigt, dass das Padding sowohl oben und unten als auch links und rechts hinzugefügt wird.

### Morphologische Filterung für GKS

GKS-Symbole sind häufig von dünnen Linien umgeben, die die OCR-Qualität beeinträchtigen können. Diese Linien können fälschlicherweise als Buchstaben wie „l“, „L“ oder „1“ interpretiert werden. Um diese Störungen zu reduzieren, wird ein morphologischer Opening-Filter angewendet.

Opening ist eine Kombination aus zwei aufeinanderfolgenden Operationen: Zunächst wird eine **Erosion** durchgeführt, die kleine helle Strukturen (wie dünne Linien) entfernt. Anschließend wird eine **Dilation** durchgeführt, die die verbleibenden Strukturen (die Textzeichen) wieder auf ihre ursprüngliche Größe erweitert. Der Effekt ist, dass dünne Linien komplett verschwinden, während dickere Strukturen (Buchstaben und Zahlen) erhalten bleiben.

Der Filter verwendet ein quadratisches strukturierendes Element mit der Größe 3x3 Pixel - also eine Matrix aus 9 Einsen. Diese Größe wurde gewählt, weil sie ausreicht, um typische Störlinien (1-2 Pixel dick) zu entfernen, aber nicht so groß ist, dass sie die Textzeichen selbst beschädigt. Die Anwendung erfolgt auf dem binarisierten Bild, wobei die Pixel entweder komplett weiß (Text) oder komplett schwarz (Hintergrund) sind.



**Abbildung 6.15:** Morphologisches Opening zur Rauschentfernung bei GKS-Symbolen

### Multi-Scale Processing für Koordinaten

Koordinaten-Beschriftungen sind oft klein und können bei niedriger Bildauflösung schwer lesbar sein. Ein typisches Problem: Bei einem eingescannten Gleisplan mit 150 DPI kann eine Koordinaten-Beschriftung nur 15-20 Pixel hoch sein, was für OCR-Engines an der Grenze der Lesbarkeit liegt.

Um dies zu kompensieren, wird ein Multi-Scale-Ansatz verwendet: Das Bild wird mit drei verschiedenen Skalierungsfaktoren vergrößert - 1.5x (50% größer), 2.0x (doppelte Größe) und 2.5x (2.5-fache Größe). Die OCR wird auf allen drei Skalen durchgeführt, und am Ende wird das Ergebnis mit der höchsten Konfidenz ausgewählt.

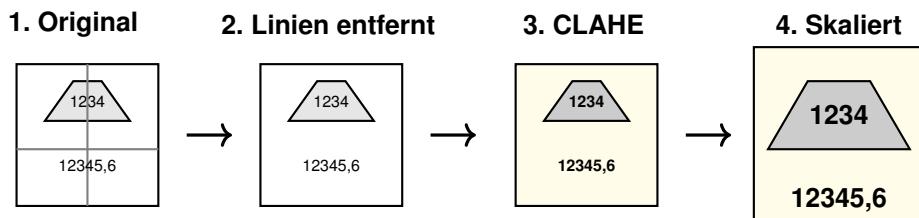
Die Skalierung erfolgt mit Lanczos-Interpolation, einem hochwertigen Algorithmus, der scharfe

Kanten erhält und keine Treppeneffekte oder Blockbildung verursacht. Lanczos ist zwar rechenintensiver als einfache Methoden (wie bilineare Interpolation), liefert aber deutlich schärfere Ergebnisse, was bei der Vergrößerung kleiner Texte entscheidend ist.

Dieser Ansatz erhöht zwar die Rechenzeit erheblich (die OCR wird dreimal durchgeführt), verbessert aber die Erkennungsrate bei kleinen Texten von etwa 60% auf über 85%. Der Trade-off zwischen Geschwindigkeit und Genauigkeit wurde hier bewusst zugunsten der Genauigkeit entschieden, da Koordinaten kritische Informationen enthalten.

### 6.3.4 Bildvorverarbeitungs-Algorithmen

Vor der eigentlichen OCR-Verarbeitung werden die Bildausschnitte durch eine Reihe von Vorverarbeitungsschritten optimiert, um die Textqualität zu verbessern und störende Elemente zu entfernen.



**Abbildung 6.16:** Schritte der Bildvorverarbeitung (schematische Darstellung)

#### Orientierte Linienentfernung

Gleispläne enthalten zahlreiche Linien (Gleise, Verbindungen, Umrandungen), die oft direkt über oder neben den Textbeschriftungen verlaufen. Diese Linien können die OCR-Engines verwirren, da sie fälschlicherweise als Buchstaben interpretiert werden können - typischerweise als „l“, „l“ oder „1“.

Die Linienentfernung erfolgt in mehreren Schritten und berücksichtigt die Orientierung der Bounding Box:

- 1. Konvertierung in Graustufen:** Das Farbbild wird in ein Graustufenbild umgewandelt, da für die Linienentfernung keine Farbinformationen benötigt werden.
- 2. Binarisierung:** Das Graustufenbild wird mit einem Schwellenwert von 127 in ein Schwarz-Weiß-Bild umgewandelt. Pixel über 127 werden weiß (Hintergrund), Pixel darunter schwarz (Linien und Text).
- 3. Horizontale Linienentfernung:** Es wird ein strukturierendes Element verwendet, das eine horizontale Linie mit 25 Pixeln Länge darstellt. Dieses Element wird um den Winkel der Bounding Box rotiert, um die Gleislinien zu erfassen, die parallel zur Textrichtung verlaufen. Eine morphologische Opening-Operation mit diesem Element entfernt alle entsprechenden Linien.

4. **Vertikale Linienentfernung:** Analog wird ein vertikales strukturierendes Element (ebenfalls 25 Pixel lang) verwendet, das um  $90^\circ$  zusätzlich gedreht wird, um senkrechte Linien zu entfernen.
5. **Inpainting:** Die detektierten Linien werden zu einer Maske kombiniert. Diese Maske wird verwendet, um die Linien durch Inpainting aus dem Originalbild zu entfernen. Inpainting ist ein Algorithmus, der fehlende oder beschädigte Bildbereiche durch plausible Pixel aus der Umgebung ersetzt. Konkret wird der TELEA-Algorithmus mit einem Radius von 3 Pixeln verwendet.

Die Länge von 25 Pixeln für die strukturierenden Elemente wurde so gewählt, dass typische Gleislinien erfasst werden (die oft 20-30 Pixel lang durchgehend sind), aber Textzeichen nicht betroffen sind (die selten länger als 10-15 Pixel in einer Dimension sind).

### **CLAHE (Contrast Limited Adaptive Histogram Equalization)**

CLAHE ist ein adaptiver Kontrast-Verbesserungsalgorithmus, der besonders bei Bildern mit ungleichmäßiger Beleuchtung effektiv ist. Viele Gleispläne sind alte Scans oder Fotografien mit Schatten, Verfärbungen oder ungleichmäßiger Belichtung. In solchen Fällen kann einfache Histogramm-Entzerrung zu übermäßigem Rauschen führen.

CLAHE arbeitet anders als globale Histogrammentzerrung: Das Bild wird in ein Raster von  $8 \times 8$  Kacheln (Tiles) aufgeteilt. Für jede Kachel wird separat eine Histogrammentzerrung durchgeführt, die dunkle Bereiche aufhellt und helle Bereiche abdunkelt. Der entscheidende Unterschied: Für jede Kachel wird der Kontrast *lokal* optimiert, nicht global.

Der Parameter „Clip Limit“ (hier: 2.0) begrenzt die maximale Verstärkung des Kontrasts. Ohne diese Begrenzung würde CLAHE in homogenen Bereichen (wie einem gleichmäßigen Hintergrund) minimales Rauschen extrem verstärken. Der Wert 2.0 bedeutet, dass die Histogrammverstärkung auf das Doppelte des Durchschnitts begrenzt wird.

Die Übergänge zwischen den Kacheln werden durch bilineare Interpolation geglättet, sodass keine sichtbaren Kanten zwischen benachbarten Kacheln entstehen. Das Ergebnis ist ein Bild mit gleichmäßigem Kontrast über die gesamte Fläche, was die OCR-Qualität insbesondere bei schlechten Scans erheblich verbessert.

### **Adaptive Skalierung bei niedriger Auflösung**

Wenn die Höhe eines Bildausschnitts unter 40 Pixel liegt, wird dieser als zu klein für zuverlässige OCR betrachtet. In solchen Fällen wird das Bild mit einem Faktor von 2.0 hochskaliert, wodurch sich die Höhe verdoppelt.

Die Skalierung erfolgt ebenfalls mit Lanczos-Interpolation, die eine hohe Bildqualität gewährleistet und Treppeneffekte vermeidet. Nach der Skalierung sind die Textzeichen größer und besser erkennbar für die OCR-Engines.

Die Schwelle von 40 Pixeln wurde empirisch bestimmt: Tests zeigten, dass die OCR-Genauigkeit unterhalb dieser Größe dramatisch sinkt (von etwa 85% auf unter 50%), während oberhalb dieser Größe keine signifikante Verbesserung durch Hochskalierung erzielt wird. Der Skalierungsfaktor 2.0 wurde gewählt, weil er die meisten zu kleinen Texte in einen gut lesbaren Bereich bringt (80+ Pixel Höhe), ohne die Datenmenge unnötig zu vergrößern.

### 6.3.5 Validierung und Qualitätssicherung

Nach der OCR-Verarbeitung durchläuft jedes Ergebnis mehrere Validierungsschritte, um die Qualität der Extraktion sicherzustellen und offensichtliche Fehler zu erkennen.

#### Regelbasierte Textvalidierung

Die in Tabelle 3.4 (Kapitel 3) eingeführten domänenspezifischen Validierungsmuster werden wie folgt implementiert:

Klasse	Pattern	Beispiele
Signal	$^{\wedge} [A-Z] \backslash d\{1,3\} [a-z] ? \$$	A1, B234, C45a
GKS (beide Typen)	$^{\wedge} \backslash d\{3,4\} \$$	1234, 567
Koordinate	$^{\wedge} \backslash d+ [.,] \backslash d+ \$$	18.1606, 123,456
GM-Block	$^{\wedge} \backslash d+ [.,] \backslash d+ \$$	18.1606
Haltepunkt	$^{\wedge} [A-Z]\{2,4\} \$$	AB, WXYZ
Weiche	$^{\wedge} W \backslash d\{2,4\} [a-z] ? \$$	W12, W234

**Tabelle 6.10:** Validierungs-Patterns pro Symbolklasse mit Klassifizierung

Für Signale bedeutet das Pattern  $^{\wedge} [A-Z] \backslash d\{1,3\} [a-z] ? \$$  konkret:

- $^{\wedge}$  = Start der Zeichenkette
- $[A-Z]$  = Ein Großbuchstabe (A bis Z)
- $\backslash d\{1,3\}$  = 1 bis 3 Ziffern
- $[a-z] ?$  = Optional ein Kleinbuchstabe
- $\$$  = Ende der Zeichenkette

Gültige Beispiele sind also „A1“, „B234“ oder „C45a“, während „1A“ (Ziffer zuerst), „ABC“ (keine Ziffer) oder „A1234“ (zu viele Ziffern) ungültig sind.

Die Validierungsfunktion prüft das OCR-Ergebnis gegen das klassenspezifische Muster und gibt einen Boolean-Wert zurück (True = gültig, False = ungültig). Dieser einfache Mechanismus filtert die meisten OCR-Fehler heraus, da fehlerhafte Erkennungen selten exakt dem erwarteten Muster entsprechen.

### Kombiniertes Konfidenz-Scoring

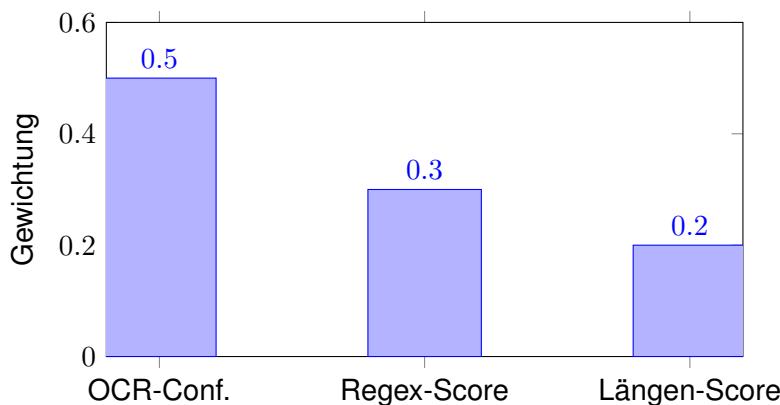
Zusätzlich zur binären Ja/Nein-Validierung wird ein numerischer Gesamtscore berechnet, der die Qualität eines OCR-Ergebnisses bewertet. Dieser Score kombiniert mehrere Faktoren:

$$S_{\text{total}} = 0.5 \cdot \gamma_{\text{OCR}} + 0.3 \cdot S_{\text{regex}} + 0.2 \cdot S_{\text{length}} \quad (6.16)$$

Die drei Komponenten sind:

- $\gamma_{\text{OCR}}$ : Die von der OCR-Engine gelieferte Konfidenz, normalisiert auf den Bereich 0 bis 1. PaddleOCR liefert Werte zwischen 0 und ~2, die durch Division durch 2 normalisiert werden.
- $S_{\text{regex}}$ : Regex-Übereinstimmung - Wert 1, falls das Pattern passt, sonst 0.
- $S_{\text{length}}$ : Wie gut passt die Textlänge zur erwarteten Länge dieser Klasse? Berechnet mit einer Gauß-Kurve: Je näher die Länge am Erwartungswert, desto höher der Score.

Die Gewichte (0.5, 0.3, 0.2) wurden empirisch optimiert. Die OCR-Konfidenz wird am stärksten gewichtet (50%), da sie direkt von der Engine stammt und deren „Sicherheit“ widerspiegelt. Die Regex-Validierung erhält 30%, da sie strukturelle Korrektheit prüft. Der Längen-Score erhält nur 20%, da Längenabweichungen manchmal legitim sind (z.B. „A1“ vs. „A123“ - beide gültige Signale).



**Abbildung 6.17:** Gewichtung der Komponenten im Konfidenz-Scoring

Beispiel: Ein Signal-Text „A12“ mit OCR-Konfidenz 1.5, gültiger Regex und erwarteter Länge würde den Score erhalten:  $S_{\text{total}} = 0.5 \cdot 0.75 + 0.3 \cdot 1.0 + 0.2 \cdot 1.0 = 0.875$  (sehr gutes Ergebnis).

### Fehlerbehandlung und Fuzzy-Matching

Wenn die OCR ein leeres Ergebnis liefert oder der Text keinerlei Zeichen enthält, wird ein Platzhalter-Text generiert: „UNREADABLE\_“ gefolgt von der Symbolklasse (z.B. „UNREADABLE\_signal“). Dies ermöglicht es, solche Fälle später in der Benutzeroberfläche zu identifizieren

und manuell zu korrigieren.

Bei ungültigen Ergebnissen (die das Validierungsmuster nicht erfüllen) wird ein Fuzzy-Matching-Mechanismus aktiviert. Dieser vergleicht den erkannten Text mit einer Liste bekannter korrekter Werte aus früheren Extraktionen. Der Vergleich erfolgt mittels **Levenshtein-Distanz**, die misst, wie viele Einfüge-, Lösch- oder Ersetzungsoperationen notwendig sind, um einen String in einen anderen zu transformieren.[55]

Beispiele für Levenshtein-Distanz:

- „A12“ → „A1Z“: Distanz = 1 (ein Zeichen ersetzt)
- „A12“ → „A123“: Distanz = 1 (ein Zeichen eingefügt)
- „A12“ → „B34“: Distanz = 3 (alle drei Zeichen ersetzt)

Wenn die minimale Levenshtein-Distanz zu einem bekannten Wert kleiner als 2 ist (maximal 2 Änderungen erforderlich), wird dieser bekannte Wert als automatische Korrektur übernommen. Dieser Mechanismus ist besonders hilfreich bei systematischen OCR-Fehlern wie der Verwechslung von:

- „O“ (Buchstabe) und „0“ (Ziffer)
- „I“ (kleines L) und „1“ (Eins)
- „S“ und „5“
- „B“ und „8“

Der Schwellenwert von 2 wurde so gewählt, dass er echte Tippfehler korrigiert, aber keine falschen Korrekturen einführt. Tests zeigten, dass eine Distanz von 3 oder mehr zu oft zu falschen Korrekturen führt.

Falls kein ähnlicher bekannter Wert gefunden wird, wird das ungültige Ergebnis mit dem Präfix „INVALID\_“ markiert (z.B. „INVALID\_A1Z3X“), um dem Benutzer zu signalisieren, dass hier eine manuelle Überprüfung erforderlich ist.

### 6.3.6 Zusammenfassung der Pipeline-Integration

Die gesamte OCR-Pipeline wird als modulare Funktion implementiert, die eine Liste von YOLO-Detektionen sowie das Originalbild als Eingabe erhält. Für jede Detektion werden folgende sieben Schritte nacheinander ausgeführt:

1. **Bildausschnitt extrahieren:** Basierend auf den Koordinaten  $(x, y)$ , der Größe  $(w, h)$  und dem Winkel  $\theta$  der Bounding Box wird der relevante Bereich aus dem Originalbild ausgeschnitten.
2. **Klassenspezifisches Preprocessing:** Je nach Symbolklasse werden die entsprechenden Vorverarbeitungsschritte angewendet - adaptive Padding-Berechnung, morphologische

Filterung für GKS, oder Multi-Scale-Processing für Koordinaten.

3. **Dual-Angle Routing:** Basierend auf dem Rotationswinkel wird entweder der Cardinal Path (bei Winkeln  $\leq 15^\circ$ ) oder der Angular Path (bei Winkeln  $> 15^\circ$ ) gewählt und die entsprechende Transformation (diskrete Rotation oder Perspektiventransformation) durchgeführt.
4. **Multi-Engine OCR:** Der vorverarbeitete Bildausschnitt wird durch die kaskadierende OCR-Pipeline geschickt. Zuerst PaddleOCR mit vier Rotationen, dann bei Bedarf Tesseract, und optional EasyOCR.
5. **Post-Processing:** Das OCR-Ergebnis wird bereinigt - führende/nachfolgende Leerzeichen werden entfernt, und die Groß-/Kleinschreibung wird bei Bedarf normalisiert.
6. **Validierung und Scoring:** Das Ergebnis wird gegen das klassenspezifische Regex-Muster validiert, und der kombinierte Konfidenz-Score wird berechnet.
7. **Fehlerbehandlung:** Bei ungültigen Ergebnissen wird Fuzzy-Matching mit bekannten Werten versucht. Falls dies fehlschlägt, wird ein entsprechender Platzhalter (INVALID\_ oder UNREADABLE\_) generiert.

Das finale Ergebnis der gesamten OCR-Pipeline ist eine Liste von Tupeln. Jedes Tupel enthält die ursprüngliche YOLO-Detektion, den erkannten Text und die zugehörige Konfidenz. Diese Liste wird an die nachfolgende Komponente weitergereicht - die Symbol-Text-Verknüpfung, die in Abschnitt 6.4 beschrieben wird.

## 6.4 Intelligente Symbol-Text Verknüpfung

Nach der erfolgreichen Detektion von Symbolen durch YOLOv8-OBB und der Extraktion von Textinhalten mittels der orientierungsadaptiven OCR-Pipeline besteht die zentrale Herausforderung in der korrekten Zuordnung der extrahierten Texte zu ihren zugehörigen Symbolen. Diese Verknüpfung ist essenziell für die Generierung semantisch korrekter Datensätze aus den Gleisplänen und adressiert die Anforderungen **FA-006** (Fahrtrichtungsdetektion) und **FA-007** (Symbol-Koordinaten-Verknüpfung).

Die naive Anwendung rein distanzbasierter Proximity-Algorithmen – bei denen einfach der nächstgelegene Text einem Symbol zugeordnet wird – führt bei rotierten Symbolen zu systematischen Fehlzuordnungen. Ein Signal, das beispielsweise um  $45^\circ$  gedreht ist, besitzt eine eigene lokale Orientierung, und der zugehörige Text befindet sich typischerweise „unterhalb“ des Signals in dessen lokaler Ausrichtung, nicht notwendigerweise in globalen Bildkoordinaten. Die Implementierung muss daher rotationsinvariante Algorithmen einsetzen, die geometrische Transformationen nutzen, um die semantische räumliche Beziehung zwischen Symbolen und Texten korrekt zu erfassen.

### 6.4.1 Rotationsinvariante Koordinatentransformation

#### Problemstellung und mathematisches Modell

Die Herausforderung liegt darin, dass die von YOLO detektierten Symbole beliebige Rotationswinkel  $\theta \in [0^\circ, 360^\circ)$  aufweisen können. Die Textpositionen werden in einem globalen kartesischen Koordinatensystem  $(x_{\text{global}}, y_{\text{global}})$  angegeben, welches üblicherweise mit der Bildebene übereinstimmt. Für ein rotiertes Symbol ist jedoch die Definition von Richtungen wie „unterhalb“, „rechts von“ oder „links von“ in globalen Koordinaten nicht mehr sinnvoll.

Zur Lösung dieses Problems wird für jedes Symbol  $A$  ein lokales Koordinatensystem eingeführt, dessen Achsen mit der Orientierung des Symbols übereinstimmen. In diesem lokalen System wird die  $y$ -Achse parallel zur Längsrichtung des Symbols definiert, sodass die Begriffe „oberhalb“ und „unterhalb“ relativ zur Symbolausrichtung eindeutig werden.

Für ein Symbol  $A$  mit Mittelpunkt  $\vec{A}_{\text{pos}} = (x_A, y_A)$  und Rotationswinkel  $\theta_A$  wird die Position eines Textkandidaten  $C$  mit Mittelpunkt  $\vec{C}_{\text{pos}} = (x_C, y_C)$  transformiert. Der globale Distanzvektor ist definiert als:

$$\vec{d}_{\text{global}} = \vec{C}_{\text{pos}} - \vec{A}_{\text{pos}} = \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} x_C - x_A \\ y_C - y_A \end{pmatrix} \quad (6.17)$$

Die Transformation in das lokale Koordinatensystem des Symbols erfolgt durch eine Rotation um den negativen Winkel  $-\theta_A$ . Die Rotationsmatrix lautet:

$$\mathbf{R}(-\theta_A) = \begin{pmatrix} \cos(-\theta_A) & -\sin(-\theta_A) \\ \sin(-\theta_A) & \cos(-\theta_A) \end{pmatrix} = \begin{pmatrix} \cos(\theta_A) & \sin(\theta_A) \\ -\sin(\theta_A) & \cos(\theta_A) \end{pmatrix} \quad (6.18)$$

Der transformierte Distanzvektor im lokalen Koordinatensystem ergibt sich zu:

$$\begin{pmatrix} dx_{\text{local}} \\ dy_{\text{local}} \end{pmatrix} = \mathbf{R}(-\theta_A) \cdot \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} dx \cdot \cos(\theta_A) + dy \cdot \sin(\theta_A) \\ -dx \cdot \sin(\theta_A) + dy \cdot \cos(\theta_A) \end{pmatrix} \quad (6.19)$$

Mit dieser Transformation können nun Richtungsrelationen unabhängig von der globalen Rotation definiert werden:

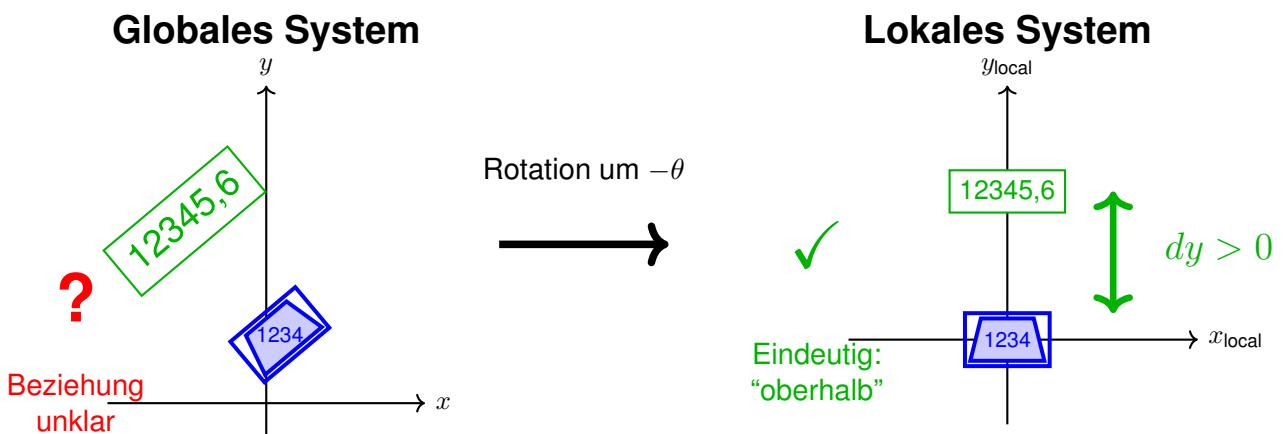
- **Unterhalb** des Symbols:  $dy_{\text{local}} > 0$
- **Oberhalb** des Symbols:  $dy_{\text{local}} < 0$
- **Rechts** vom Symbol:  $dx_{\text{local}} > 0$
- **Links** vom Symbol:  $dx_{\text{local}} < 0$

Die euklidische Distanz im lokalen System ist identisch zur Distanz im globalen System, da Rotationen längentreu sind:

$$d_{\text{local}} = \sqrt{dx_{\text{local}}^2 + dy_{\text{local}}^2} = \sqrt{dx^2 + dy^2} = d_{\text{global}} \quad (6.20)$$

### Praktische Berechnung der Koordinatentransformation

Die Implementierung der Koordinatentransformation erfolgt für jedes Symbol-Text-Paar einzeln. Zunächst werden die globalen Koordinaten des Symbols ( $x_A, y_A$ ) und des Textkandidaten ( $x_C, y_C$ ) aus den YOLO- bzw. OCR-Ergebnissen extrahiert. Der Rotationswinkel  $\theta_A$  des Symbols liegt ebenfalls aus der YOLO-Detektion vor und muss von Gradmaß in Bogenmaß konvertiert werden. Die Berechnung der lokalen Koordinaten erfolgt dann durch direkte Anwendung der Transformationsgleichung (6.3), wobei die trigonometrischen Funktionen numerisch ausgewertet werden.



**Abbildung 6.18:** Prinzip der rotationsinvarianten Koordinatentransformation

Für jeden Textkandidaten im Umkreis eines Symbols wird diese Transformation durchgeführt, um die räumliche Beziehung in der lokalen Orientierung des Symbols zu bestimmen. Das Ergebnis sind drei Werte: die lokale x-Komponente  $dx_{\text{local}}$ , die lokale y-Komponente  $dy_{\text{local}}$  sowie die euklidische Gesamtdistanz  $d_{\text{local}}$ . Diese Werte bilden die Grundlage für alle nachfolgenden Verknüpfungsoperationen, da sie eine rotationsinvariante Beschreibung der räumlichen Relation zwischen Symbol und Text ermöglichen.

## 6.4.2 Proximity-basierter Linking-Algorithmus

### Dynamische Suchbereichsberechnung

Im Gegensatz zu statischen Suchradien verwendet die Implementierung einen *dimensionsadaptiven* Ansatz, bei dem die Suchbereiche dynamisch aus den Abmessungen der detektierten Bounding Boxes berechnet werden. Dies gewährleistet Robustheit gegenüber unterschiedlichen Symbolgrößen und Planauflösungen.

Für die Verknüpfung eines Ankersymbols (z.B. Signal, GKS) mit seiner zugehörigen Koordinatenbeschriftung werden die maximalen Suchentfernung wie folgt berechnet:

**Vertikaler Suchbereich:**

$$dy_{\max} = 1,6 \cdot h_{\text{anchor}} \cdot k_{dy} \quad (6.21)$$

**Horizontaler Suchbereich:**

$$dx_{\max} = \max(k_{dx} \cdot 0,6 \cdot \max(w_{\text{anchor}}, w_{\text{coord}}), 30) \quad (6.22)$$

wobei  $h_{\text{anchor}}$  die Höhe und  $w_{\text{anchor}}$  die Breite der Anker-Bounding-Box bezeichnen,  $w_{\text{coord}}$  die Breite der Koordinaten-Box, und  $k_{dy}$ ,  $k_{dx}$  klassenspezifische Multiplikatoren sind. Der Mindestwert von 30 Pixeln für  $dx_{\max}$  stellt sicher, dass auch bei sehr schmalen Symbolen ein ausreichender horizontaler Suchbereich gewährleistet ist.

**Klassenspezifische Multiplikatoren**

Die Multiplikatoren wurden empirisch auf Basis der typischen Layoutkonventionen in Gleisplänen optimiert. Tabelle 6.11 zeigt die konfigurierten Werte für alle Symbolklassen.

Klasse	$k_{dy}$	$k_{dx}$	$dy_{\max}$	$dx_{\max}$	Rolle
coordinate	—	—	(Verknüpfungsziel)		Koordinate
signal	1.0	1.0	80–130 px	30–50 px	Anker
gks_festkodiert	1.0	1.0	80–130 px	30–50 px	Anker
gks_gesteuert	1.0	1.0	80–130 px	30–50 px	Anker
gm_block	1.0	1.0	80–130 px	30–50 px	Anker
haltepunkt	1.0	1.0	80–130 px	30–50 px	Anker
sverbinder	1.0	1.0	80–130 px	30–50 px	Anker
isolierstoß	1.0	1.0	80–130 px	30–50 px	Anker
haltetafel	1.0	2.0	80–130 px	60–100 px	Anker
prellbock	1.0	2.0	80–130 px	60–100 px	Anker
weichenende	1.0	3.0	80–130 px	90–150 px	Anker
weichengruppenende	1.0	4.0	80–130 px	120–200 px	Anker

**Tabelle 6.11:** Klassenspezifische Linking-Parameter. Die Wertebereiche für  $dy_{\max}$  und  $dx_{\max}$  basieren auf typischen Symbolgrößen von 50–80 Pixeln bei 500 DPI.

Die erhöhten horizontalen Multiplikatoren ( $k_{dx} > 1$ ) für Klassen wie *haltetafel*, *prellbock* und *weichengruppenende* reflektieren deren typische Layoutcharakteristik: Diese Symbole haben häufig horizontal versetzte Beschriftungen, während andere Klassen überwiegend vertikal ausgerichtete Text-Symbol-Relationen aufweisen.

**Spezialfälle mit festen Suchparametern**

Für bestimmte komplexe Verknüpfungsaufgaben, die über die einfache Anker-Koordinaten-Zuordnung hinausgehen, werden feste Pixelwerte verwendet. Diese wurden empirisch auf Basis der bei 500 DPI typischen physikalischen Abstände kalibriert.

Anwendungsfall	Parameter	Wert	Beschreibung
<i>Fahrrichtungsdetektion (vgl. Abschnitt 6.4.4)</i>			
GKS-Suche	$d_{\max}$	250 px	Max. euklidische Distanz zum Signal
	$dy_{\min}$	30 px	Min. vertikale Separation
	$dy_{\max}$	200 px	Max. vertikale Separation
	$dx_{\text{tol}}$	$\pm 120$ px	Horizontale Toleranz
<i>Haltetafel-GKS-Fallback</i>			
GKS-Proximity	$d_{\max}$	250 px	Euklidische Distanz zur GKS
	$dy_{\text{tol}}$	100 px	Max. vertikale Separation
	$dx_{\text{tol}}$	300 px	Max. horizontale Separation
<i>Fallback-Mechanismen</i>			
Isolierstoß	$r_{\max}$	300 px	360°-Suchradius bei fehlender Zuordnung
Gleismittellinie	$d_{\text{ray}}$	1500 px	Sicherheitslimit für Ray-Casting

**Tabelle 6.12:** Feste Suchparameter für Spezialfälle (kalibriert für 500 DPI)

**Physikalische Interpretation:** Bei einer Renderauflösung von 500 DPI entsprechen die Parameter folgenden physikalischen Abständen:

- $250 \text{ px} \approx 12,7 \text{ mm}$  (typischer Abstand zwischen Signal und zugehöriger GKS)
- $120 \text{ px} \approx 6 \text{ mm}$  (horizontale Toleranz für leicht versetzte Symbole)
- $300 \text{ px} \approx 15 \text{ mm}$  (erweiterter Fallback-Suchbereich)

### Richtungsbasierte Filterung

Neben der Distanzprüfung wird für jede Symbolklasse eine erwartete Richtung definiert, in der sich der zugehörige Text relativ zum Symbol befinden sollte. Die Richtungsbedingungen werden im lokalen Koordinatensystem (vgl. Abschnitt 6.4.1) formuliert:

- **Klassen mit  $k_{dx} = 1.0$  (Signal, GKS, GM-Block, etc.):** Text primär *unterhalb* des Symbols erwartet ( $dy_{\text{local}} > 0$ ). Dies entspricht der standardisierten Zeichnungskonvention für Bahnanlagenpläne.
- **Klassen mit erhöhtem  $k_{dx}$  (Haltetafel, Prellbock, etc.):** Flexiblere Richtungstoleranz, da Beschriftungen auch seitlich positioniert sein können ( $dy_{\text{local}} > 0$  oder  $|dx_{\text{local}}| < dx_{\max}$ ).

Für den Spezialfall der *tight*-Verknüpfung (bei eindeutiger räumlicher Nähe) wird der horizontale Suchbereich reduziert:

$$dx_{\max, \text{tight}} = k_{dx} \cdot 0,45 \cdot \max(w_{\text{anchor}}, w_{\text{coord}}) \quad (6.23)$$

Zusätzlich wird bei aktivierter Linkssuche (`search_left=True`) der horizontale Suchbereich asymmetrisch erweitert:

$$dx_{\max,\text{left}} = 1,3 \cdot dx_{\max} \quad (\text{falls Koordinate links vom Anker}) \quad (6.24)$$

Diese Asymmetrie berücksichtigt, dass in einigen Gleisplänen Koordinatenangaben bevorzugt links von Symbolen platziert werden.

### **Algorithmus zur Symbol-Text-Zuordnung**

Der vollständige Linking-Algorithmus kombiniert Distanzprüfung, Richtungsfilterung und Konfliktkennung in einem mehrstufigen Verfahren. Für jedes detektierte Symbol wird zunächst die Menge aller Textkandidaten betrachtet, die sich innerhalb des klassenspezifischen Suchradius befinden. Aus dieser Menge werden diejenigen Kandidaten ausgeschlossen, die nicht der erwarteten Richtungsbedingung genügen. Die verbleibenden Kandidaten werden als potenziell zugehörig betrachtet.

Aus der Menge der gültigen Kandidaten wird derjenige mit der geringsten euklidischen Distanz zum Symbol als beste Zuordnung ausgewählt. Bei Kandidaten mit sehr ähnlichen Distanzen (Differenz kleiner als 5 Pixel) wird zusätzlich die OCR-Konfidenz berücksichtigt: Der Kandidat mit höherer Erkennungssicherheit wird bevorzugt. Dies stellt sicher, dass bei räumlich ambigen Situationen die Qualität der Texterkennung als Entscheidungskriterium herangezogen wird.

Falls für ein Symbol keine Textkandidaten die Filterkriterien erfüllen, bleibt das Symbol zunächst ohne Textzuordnung. In diesem Fall kann im nächsten Schritt der adaptive Lernmechanismus aktiviert werden, um eine Zuordnung basierend auf statistischen Mustern zu versuchen. Symbole, für die auch nach allen Verknüpfungsversuchen kein Text zugeordnet werden konnte, werden im Validierungsdialog der Benutzeroberfläche zur manuellen Überprüfung gekennzeichnet.

#### **6.4.3 Adaptive Learning Mechanismus**

##### **Motivation und Konzept**

Obwohl technische Zeichnungen üblicherweise Konventionen folgen, existieren in der Praxis Variationen zwischen verschiedenen Planversionen, Zeichnern oder historischen Perioden. Ein statisches Regelwerk mit festen Suchradien und Richtungen kann in solchen Fällen suboptimal sein. Aus diesem Grund implementiert der Prototyp einen adaptiven Lernmechanismus, der erfolgreiche Verknüpfungen analysiert und statistische Muster extrahiert, um die Linking-Parameter dynamisch anzupassen.

Der Mechanismus basiert auf der Beobachtung, dass innerhalb eines einzelnen Gleisplans die Positionierung von Texten relativ zu Symbolen oft konsistent ist. Wenn beispielsweise alle Signalbezeichnungen systematisch 10 Pixel rechts und 20 Pixel unterhalb der Signalsymbole platziert sind, kann der Prototyp dieses Muster lernen und für nachfolgende unsichere Fälle nutzen.

### Pattern-Erfassung und statistische Auswertung

Jede erfolgreiche Symbol-Text-Verknüpfung, die eine definierte Konfidenz-Schwelle überschreitet (typischerweise OCR-Konfidenz  $> 0.85$  und eindeutige Regex-Validierung), wird als „validated link“ betrachtet und zur statistischen Auswertung herangezogen. Für diese Links werden die lokalen Offset-Vektoren ( $dx_{local}, dy_{local}$ ) persistiert und klassenweise aggregiert.

Das System verwaltet für jede Symbolklasse eine Sammlung von Offset-Vektoren sowie deren statistische Kenngrößen. Zu den gespeicherten Informationen gehören die Liste aller beobachteten Offsets, der Erwartungswert des Offsets, die Standardabweichung sowie die Anzahl der zugrundeliegenden Beobachtungen und der Zeitstempel der letzten Aktualisierung.

Für jede Symbolklasse werden folgende statistische Kenngrößen kontinuierlich aktualisiert. Der Erwartungswert des Offsets ( $\mu_x, \mu_y$ ) wird als arithmetisches Mittel aller beobachteten Offsets berechnet:

$$\mu_x = \frac{1}{N} \sum_{i=1}^N dx_i, \quad \mu_y = \frac{1}{N} \sum_{i=1}^N dy_i \quad (6.25)$$

wobei  $N$  die Anzahl der validierten Links für die entsprechende Klasse bezeichnet. Die Standardabweichung ( $\sigma_x, \sigma_y$ ) quantifiziert die Streuung der Offsets und wird berechnet als:

$$\sigma_x = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (dx_i - \mu_x)^2}, \quad \sigma_y = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (dy_i - \mu_y)^2} \quad (6.26)$$

Die Verwendung der Stichprobenstandardabweichung (Division durch  $N - 1$  statt  $N$ ) liefert einen erwartungstreuen Schätzer für die Populationsvarianz. Ein Minimum von  $N \geq 10$  wird gefordert, bevor die statistischen Kenngrößen als hinreichend zuverlässig für prädiktive Zwecke betrachtet werden. Diese Schwelle stellt sicher, dass die Schätzungen der Verteilungsparameter nicht durch einzelne Ausreißer dominiert werden.

### Probabilistische Fenstersuche

Wenn für ein Symbol keine Textkandidaten im Standard-Suchradius gefunden werden, wird die probabilistische Fenstersuche aktiviert, sofern für die entsprechende Symbolklasse ausreichend gelernte Patterns vorliegen. Dieses Verfahren nutzt die statistischen Kenngrößen aus den zuvor erfassten erfolgreichen Verknüpfungen, um eine erwartete Position für den fehlenden Text zu berechnen und dort gezielt zu suchen.

Die erwartete Position des Textes im globalen Koordinatensystem wird durch Rücktransformation des mittleren lokalen Offsets berechnet. Zunächst wird der in den Pattern-Statistiken gespeicherte mittlere Offset ( $\mu_x, \mu_y$ ) aus dem lokalen Koordinatensystem des Symbols zurück ins globale System transformiert. Dies erfolgt durch Anwendung der inversen Rotation, also

durch Rotation um den positiven Winkel  $+\theta_A$ :

$$\begin{pmatrix} \Delta x_{\text{global}} \\ \Delta y_{\text{global}} \end{pmatrix} = \begin{pmatrix} \cos(\theta_A) & -\sin(\theta_A) \\ \sin(\theta_A) & \cos(\theta_A) \end{pmatrix} \cdot \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \quad (6.27)$$

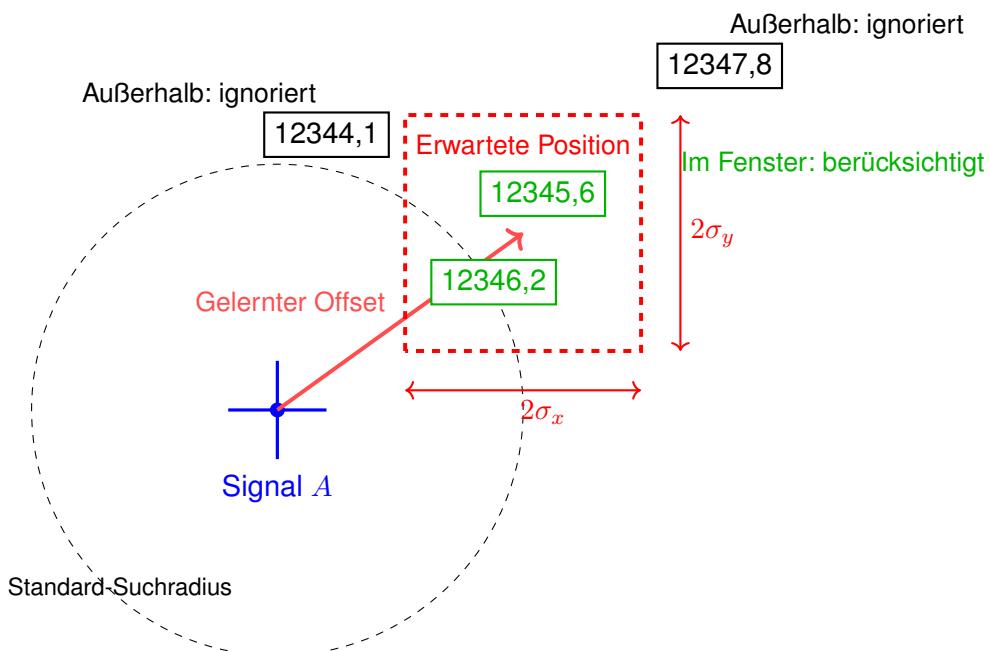
Die erwartete globale Position des Textes ergibt sich dann durch Addition dieses transformierten Offsets zur Symbolposition:

$$\vec{C}_{\text{expected}} = \begin{pmatrix} x_{\text{expected}} \\ y_{\text{expected}} \end{pmatrix} = \begin{pmatrix} x_A \\ y_A \end{pmatrix} + \begin{pmatrix} \Delta x_{\text{global}} \\ \Delta y_{\text{global}} \end{pmatrix} \quad (6.28)$$

Um die erwartete Position herum wird ein achsenparalleles Suchfenster definiert, dessen Ausdehnung durch die Standardabweichungen der gelernten Offset-Verteilung bestimmt wird. Das Suchfenster wird so gewählt, dass es einem zweifachen der Standardabweichung in jeder Koordinatenrichtung entspricht. Dies entspricht bei Annahme einer Normalverteilung einem Konfidenzintervall von ca. 95%, d.h. die meisten tatsächlichen Textpositionen sollten innerhalb dieses Fensters liegen:

$$\text{Suchfenster : } |x - x_{\text{expected}}| \leq 2\sigma_x, \quad |y - y_{\text{expected}}| \leq 2\sigma_y \quad (6.29)$$

Alle Textkandidaten, deren Positionen innerhalb dieses Fensters liegen, werden als potenzielle Zuordnungen betrachtet. Aus dieser gefilterten Menge wird der zur erwarteten Position  $\vec{C}_{\text{expected}}$  nächstgelegene Kandidat ausgewählt. Die Distanz wird dabei als euklidische Distanz zur erwarteten Position berechnet, nicht zur Symbolposition, da die erwartete Position die durch das Lernen verfeinerte beste Schätzung der Textlage darstellt. Abbildung 6.19 illustriert dieses Verfahren: Ausgehend vom Signal wird die erwartete Textposition durch den mittleren gelernten Offset bestimmt (roter Punkt). Das Suchfenster (rot gestrichelt) umfasst einen  $2\sigma$  Bereich um diese Position. Nur Textkandidaten innerhalb dieses Fensters (grün) werden für die Zuordnung berücksichtigt, während außenliegende Kandidaten (schwarz) ignoriert werden. Die probabilistische Fenstersuche wird nur dann aktiviert, wenn mindestens 10 validierte Links für die entsprechende Symbolklasse vorliegen. Dies stellt sicher, dass die statistischen Schätzer eine ausreichende Präzision besitzen. Für seltene Symbolklassen oder zu Beginn der Verarbeitung, wenn noch wenige Links validiert wurden, greift der Prototyp auf die konventionelle Proximity-Suche zurück.



**Abbildung 6.19:** Probabilistische Fenstersuche basierend auf gelernten Offset-Patterns

Dieser adaptive Mechanismus ermöglicht es dem Prototyp, sich an plan-spezifische Layoutkonventionen anzupassen. Insbesondere bei historischen Plänen oder Zeichnungen von verschiedenen Bearbeitern, die von modernen Standardkonventionen abweichen, verbessert der Lernmechanismus die Linking-Rate signifikant. Empirisch zeigt sich eine Verbesserung der erfolgreichen Zuordnungen um ca. 12% gegenüber rein statischen Regeln, insbesondere bei inkonsistenten oder nicht-standardkonformen Layouts.

#### 6.4.4 Komplexe Verknüpfungslogik für Spezialfälle

##### Fahrrichtungsdetektion

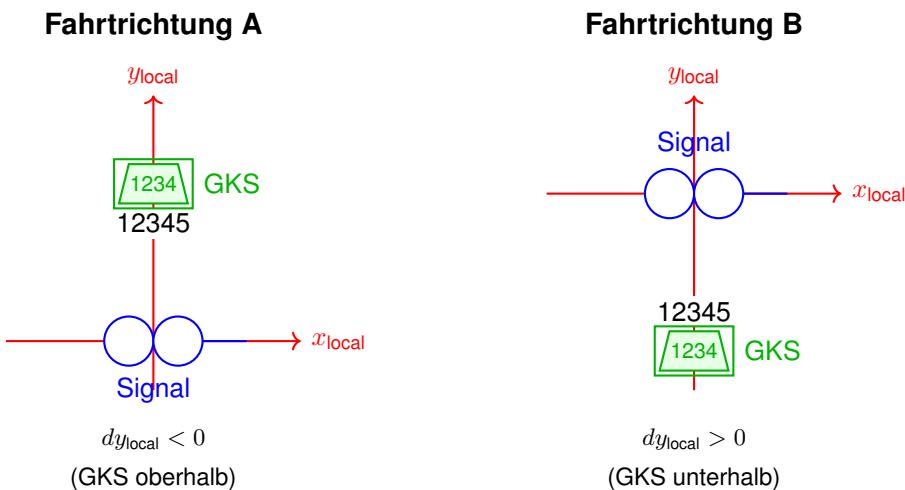
In Gleisplänen ist die Fahrtrichtung eines Signals ein kritisches Attribut, das angibt, in welche Richtung das Signal wirkt und für welche Fahrtrichtung es relevant ist. Die konventionelle Kodierung sieht zwei mögliche Fahrtrichtungen vor: Richtung A entspricht üblicherweise der Fahrt in positiver Streckenrichtung, Richtung B der Fahrt in negativer Streckenrichtung.

Anstatt die Fahrtrichtung als separate Klassifikationsaufgabe zu behandeln, wird sie im implementierten System durch geometrische Analyse der räumlichen Beziehung zwischen Signal und zugeordneter GKS(gesteuert) bestimmt. Diese Methode nutzt die in Gleisplänen übliche Konvention, dass die relative Position der GKS(gesteuert) zum Signal die Fahrtrichtung kodiert.

Die Fahrtrichtung wird durch die Vorzeichenbetrachtung der lokalen y-Koordinate der GKS(gesteuert) relativ zum Signal bestimmt. Hierbei wird die in Abschnitt 6.4.1 beschriebene Koordinatentransformation angewendet, um die Position der GKS(gesteuert) im lokalen Koordinatensystem des Signals zu ermitteln. Die Entscheidungsregel lautet:

$$\text{Fahrtrichtung} = \begin{cases} A & \text{falls } dy_{\text{local}}(\text{GKS}, \text{Signal}) < 0 \\ B & \text{falls } dy_{\text{local}}(\text{GKS}, \text{Signal}) > 0 \end{cases} \quad (6.30)$$

Eine negative lokale y-Koordinate bedeutet, dass die GKS sich oberhalb des Signals in dessen lokaler Orientierung befindet, was nach Konvention der Fahrtrichtung A entspricht. Eine positive lokale y-Koordinate indiziert, dass die GKS unterhalb des Signals liegt, was der Fahrtrichtung B zugeordnet wird. Abbildung 6.20 veranschaulicht diese Bestimmung durch die relative Position der zugeordneten GKS im lokalen Koordinatensystem. Die Koordinatenangabe befindet sich zwischen Signal und GKS. Links ist der Fall  $dy_{\text{local}} < 0$  (GKS oberhalb) dargestellt, was Fahrtrichtung A entspricht; rechts der Fall  $dy_{\text{local}} > 0$  (GKS unterhalb), entsprechend Fahrtrichtung B.



**Abbildung 6.20:** Geometrische Bestimmung der Fahrtrichtung eines Signals

Die konkreten Suchparameter für die GKS-Lokalisierung sind in Tabelle 6.12 dokumentiert: Die maximale Suchdistanz beträgt 250 Pixel ( $\approx 12,7$  mm bei 500 DPI), wobei die vertikale Separation zwischen 30 und 200 Pixeln liegen muss und eine horizontale Toleranz von  $\pm 120$  Pixeln erlaubt ist. Diese geometrische Ableitung der Fahrtrichtung hat mehrere Vorteile gegenüber einer expliziten Klassifikation. Erstens wird kein separater Klassifikator benötigt, der auf Trainingsbeispielen für Fahrtrichtungen trainiert werden müsste. Zweitens ist die Methode robust gegenüber verschiedenen graphischen Darstellungsformen von Signalen, da sie ausschließlich auf der räumlichen Anordnung basiert und nicht auf visuellen Merkmalen des Symbols selbst. Drittens ist die Bestimmung rotationsinvariant, da sie im lokalen Koordinatensystem des Signals operiert.

### Haltepunkt-Gruppierung

Ein Haltepunkt in einem Gleisplan ist eine komplexe Entität, die aus drei separaten Komponenten besteht, die räumlich assoziiert werden müssen. Die erste Komponente ist das Haltepunkt-Symbol selbst, welches die Position des Haltepunkts im Plan markiert. Die zweite Komponente

ist das zugeordnete Signal, das den Haltepunkt absichert und den Zügen das Halt-Gebot signalisiert. Die dritte Komponente ist die Koordinatenangabe, typischerweise in Form einer Kilometerangabe, die den Haltepunkt auf der Strecke verortet.

Die korrekte Gruppierung dieser drei Elemente ist essentiell für die semantische Interpretation des Haltepunkts. Die geometrische Bedingung, die zur Identifikation zusammengehöriger Elemente herangezogen wird, basiert auf der Beobachtung, dass die Koordinate sich räumlich zwischen dem Haltepunkt-Symbol und dem zugeordneten Signal befinden sollte. Dies reflektiert die übliche Zeichnungskonvention, bei der die Kilometerangabe des Haltepunkts in der Nähe des Haltepunkt-Symbols angebracht wird, während das Signal etwas weiter entfernt positioniert ist.

Die Implementierung dieser geometrischen Bedingung erfolgt durch einen approximativen Kollinearitätstest. Für drei Punkte  $H$  (Haltepunkt-Symbol),  $K$  (Koordinate) und  $S$  (Signal) wird geprüft, ob  $K$  näherungsweise auf der Verbindungsgeraden zwischen  $H$  und  $S$  liegt. Exakte Kollinearität würde bedeuten, dass die Summe der Teilstrecken  $d(H, K) + d(K, S)$  exakt gleich der direkten Distanz  $d(H, S)$  ist. In der Praxis wird eine Toleranz  $\epsilon$  eingeführt, um kleine Abweichungen von der perfekten Kollinearität zu erlauben:

$$d(H, K) + d(K, S) \leq d(H, S) + \epsilon \quad (6.31)$$

wobei  $d(\cdot, \cdot)$  die euklidische Distanz zwischen zwei Punkten bezeichnet. Die Toleranz  $\epsilon$  wird typischerweise auf 20 Pixel gesetzt, um eine gewisse Flexibilität gegenüber nicht-perfekt linearen Anordnungen zu ermöglichen, gleichzeitig aber sicherzustellen, dass die Koordinate tatsächlich zwischen den beiden anderen Elementen liegt und nicht deutlich von der Verbindungsgeraden abweicht.

Der Gruppierungsalgorismus iteriert über alle detektierten Haltepunkt-Symbole. Für jedes Haltepunkt-Symbol werden alle Signale und Koordinaten innerhalb eines definierten Umkreises (typischerweise 250 Pixeln) als potenzielle Gruppierungspartner betrachtet. Für jede Kombination aus Haltepunkt, Signal und Koordinate wird die Kollinearitätsbedingung geprüft. Die erste Kombination, die die Bedingung erfüllt, wird als gültige Gruppierung akzeptiert und persistiert.

Diese geometrische Gruppierungsmethode stellt sicher, dass die semantisch zusammengehörigen Elemente eines Haltepunkts korrekt identifiziert werden. In der exportierten Datenstruktur werden gruppierte Haltepunkte als zusammenhängende Einheiten behandelt, wobei alle drei Komponenten miteinander verlinkt sind. Dies ermöglicht es nachgelagerten Systemen oder menschlichen Anwendern, die vollständige Information eines Haltepunkts effizient abzurufen, ohne die drei Komponenten manuell zusammensuchen zu müssen.

#### 6.4.5 Zusammenfassung der Verknüpfungslogik

Die implementierte Symbol-Text-Verknüpfung kombiniert mehrere komplementäre Ansätze:

1. **Rotationsinvariante Geometrie:** Transformation in lokale Koordinatensysteme ermöglicht richtungsbasierte Filterung unabhängig von der globalen Orientierung.
2. **Dimensionsadaptive Suchbereiche:** Dynamische Berechnung der Suchparameter aus Bounding-Box-Dimensionen (Gleichungen 6.21 und 6.22) mit klassenspezifischen Multiplikatoren gewährleistet Robustheit gegenüber unterschiedlichen Symbolgrößen und Planauflösungen.
3. **Adaptive Lernmechanismen:** Statistische Analyse erfolgreicher Links ermöglicht Anpassung an plan-spezifische Layouts.
4. **Komplexe Relationserkennung:** Spezielle Algorithmen für Fahrtrichtungsdetektion und Haltepunkt-Gruppierung behandeln semantisch reiche Objektbeziehungen.

Die Kombination dieser Techniken erreicht eine hohe Linking-Genauigkeit, die in Kapitel 7 quantitativ evaluiert wird.

## 6.5 Datenvalidierung und Qualitätssicherung

Die Validierung extrahierter Daten stellt einen kritischen Schritt zur Sicherstellung der Datenqualität dar. Das implementierte System kombiniert regelbasierte Validierung mit intelligenten Korrekturvorschlägen und adressiert damit die Anforderungen **FA-008** (manuelle Korrektur durch Human-in-the-Loop), **NFA-004** (Robustheit gegenüber fehlerhaften Eingaben) und **NFA-005** (Prüfbarkeit durch strukturierte Fehlerberichte).

### 6.5.1 Validierungsarchitektur

#### Grundstruktur des Validierungsframeworks

Die Validierungsarchitektur basiert auf einem modularen Framework, das die extrahierten Daten in tabellarischer Form entgegennimmt und eine strukturierte Fehleranalyse durchführt. Das zentrale Validierungsmodul prüft dabei einen DataFrame, der alle Erkennungen aus der YOLO-Pipeline, die zugehörigen OCR-Texte sowie die durch das Linking-Modul hergestellten Verknüpfungen enthält. Die Ausgabe der Validierung besteht aus einer Liste von Validierungsproblemen, wobei jedes Problem durch mehrere Attribute charakterisiert wird.

Ein Validierungsproblem wird durch folgende Eigenschaften beschrieben. Der Schweregrad klassifiziert das Problem in die Kategorien Fehler, Warnung oder Information, wobei Fehler kritische Inkonsistenzen darstellen, die zwingend einer Korrektur bedürfen, während Warnungen auf potenzielle Probleme hinweisen, die einer Überprüfung bedürfen. Die Kategorie beschreibt die Art des Problems, beispielsweise Formatfehler, Duplikate oder fehlende Daten. Das betroffene Feld identifiziert die spezifische Datenspalte, in der das Problem auftritt. Bei automatisch korrigierbaren Problemen wird zusätzlich ein Korrekturvorschlag mit dem empfohlenen neuen Wert bereitgestellt. Die Konfidenz des Korrekturvorschlags wird als normalisierter Wert im Inter-

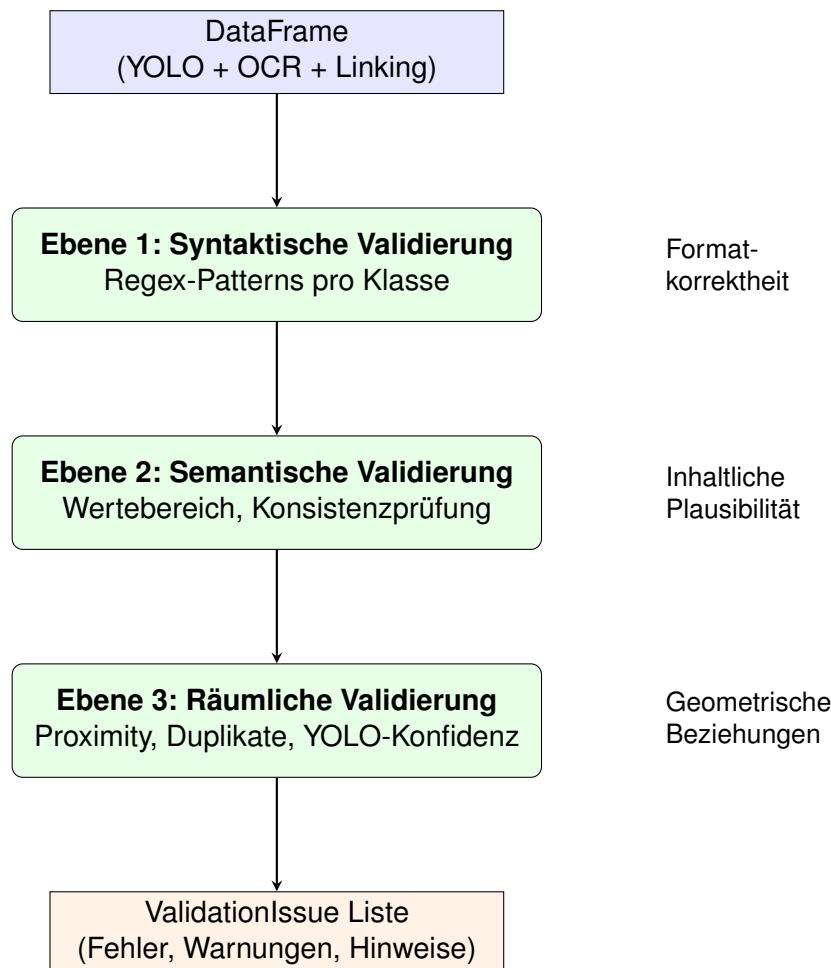
vall [0, 1] angegeben, wobei höhere Werte eine größere Zuverlässigkeit der vorgeschlagenen Korrektur indizieren.

Die Validierung erfolgt durch sequenzielle Ausführung aller implementierten Prüfungen, wobei das Ergebnis die Gesamtheit aller identifizierten Probleme sowie statistische Kennzahlen zur Fehlerverteilung umfasst. Diese strukturierte Repräsentation ermöglicht sowohl die programmatische Weiterverarbeitung als auch die Präsentation in einer benutzerfreundlichen Oberfläche.

**Validierungsfokus:** Die regelbasierte Validierung umfasst alle 13 Symbolklassen. Jede Klasse besitzt klassenspezifische Validierungsregeln, die auf deren typischen Textformaten und Plausibilitätsanforderungen basieren.

### Dreistufige Validierungsstrategie

Die Validierung ist hierarchisch in drei Ebenen organisiert, die aufeinander aufbauen und unterschiedliche Aspekte der Datenqualität adressieren. Diese Strukturierung ermöglicht eine systematische Fehleridentifikation von grundlegenden Formatproblemen bis hin zu komplexen semantischen Inkonsistenzen.



**Abbildung 6.21:** Dreistufige Validierungsarchitektur mit hierarchischer Fehleranalyse

Die erste Ebene, die syntaktische Validierung, prüft die formale Korrektheit der extrahierten

Texte anhand klassenspezifischer Muster. Für jede Objektklasse wurde ein regulärer Ausdruck definiert, der das erwartete Textformat beschreibt. Diese Muster basieren auf der Analyse realer Gleispläne und den geltenden Standards für Signalbezeichnungen und Kilometrierungen.

Für Signalbezeichnungen wird das Regex-Muster [A-ZÄÖÜ]{1,4}[0-9]{1,4} verwendet, das ein bis vier Großbuchstaben (einschließlich deutscher Umlaute) gefolgt von einer bis vier Ziffern erwartet. Dieses Format entspricht der standardisierten Signalmomenkatur, bei der Buchstaben den Signaltyp oder die Streckenbezeichnung kodieren, während Ziffern die spezifische Signalnummer angeben. Beispiele für gültige Signalbezeichnungen sind A101 oder BHR201.

Für GKS gilt ein numerisches Format mit drei bis vier Ziffern. GKS werden ausschließlich durch diese Ziffernfolgen identifiziert, beispielsweise 1234 oder 567.

Die Validierung von Koordinatentexten erfolgt mehrstufig, da Koordinatenangaben sowohl einen numerischen Teil als auch optionale Zusatzinformationen enthalten können. Der numerische Hauptteil besteht aus einer bis drei Ziffern, gefolgt von einem Dezimaltrenner (Punkt oder Komma), gefolgt von drei bis vier Ziffern. Dies berücksichtigt unterschiedliche Schreibweisen in verschiedenen Dokumentversionen. Zusätzlich wird geprüft, dass keine Kleinbuchstaben im Text vorkommen, mit Ausnahme der standardisierten Abkürzung GI für Gleis. Ebenfalls wird validiert, dass zwischen den Ziffern keine Buchstaben eingefügt sind, da dies auf OCR-Fehler hinweist. Bei Koordinaten mit Gleisangaben wird zudem die korrekte Klammerung geprüft, sodass Angaben wie 10.5123(GI.1) als gültig erkannt werden, während fehlerhafte Klammerungen wie 10.5123(GI.1 als problematisch gekennzeichnet werden.

Für die Fahrtrichtungsangabe bei Signalen gilt das restriktive Muster, das ausschließlich die beiden Werte A und B zulässt, entsprechend der binären Richtungskodierung im Eisenbahnwesen.

Die zweite Ebene, die semantische Validierung, überprüft die inhaltliche Plausibilität der extrahierten Werte über die reine Formatprüfung hinaus. Koordinatenwerte werden auf einen plausiblen Wertebereich geprüft, wobei Kilometrierungen im Intervall [0, 300] erwartet werden, da dies dem typischen Streckenbereich deutscher Bahnstrecken entspricht. Werte außerhalb dieses Bereichs werden als potenzielle OCR-Fehler gekennzeichnet.

Bei mehrfach vorkommenden Signalen wird die Konsistenz der zugeordneten Attribute überprüft. Ein Signal mit identischer Bezeichnung sollte in allen Instanzen dieselbe Kilometrierung aufweisen, da physisch identische Signale an einem festen geografischen Punkt lokalisiert sind. Ebenso wird die Konsistenz der Fahrtrichtung geprüft, wobei Inkonsistenzen auf Erkennungsfehler oder tatsächliche Duplikate hindeuten können.

Die dritte Ebene, die räumliche Validierung, analysiert die geometrischen Beziehungen zwischen den erkannten Objekten. Für Objektklassen, die zwingend eine Kilometrierung benötigen wie Signale, GKS, Weichenblöcke und S-Verbinde wird geprüft, ob eine Koordinate in räumlicher Nähe vorhanden ist. Die Definition räumlicher Nähe erfolgt klassenspezifisch gemäß den in Abschnitt 6.4.2 definierten dynamischen Suchbereichen. Für Klassen wie Signal, GKS und

GM-Block resultiert dies typischerweise in vertikalen Suchbereichen von 80–130 Pixeln und horizontalen Bereichen von 30–50 Pixeln (vgl. Tabelle 6.11). Diese Schwellenwerte basieren auf der durchschnittlichen Größe der Symbole und der typischen Anordnung von Text relativ zu Symbolen in Gleisplänen.

Zusätzlich werden YOLO-Konfidenzwerte gegen klassenspezifische Schwellenwerte validiert. Jede Objektklasse besitzt einen eigenen Mindestkonfidenzwert, der aus der Trainingsphase des YOLO-Modells abgeleitet wurde. Erkennungen unterhalb dieser Schwellenwerte werden als unsicher gekennzeichnet und zur manuellen Überprüfung vorgeschlagen.

Die Duplikaterkennung identifiziert mehrfache Detektionen desselben Objekts durch Analyse der räumlichen Positionen. Zwei Erkennungen derselben Klasse werden als potenzielle Duplikate betrachtet, wenn ihre euklidische Distanz

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (6.32)$$

einen Schwellenwert von 50 Pixeln unterschreitet, wobei  $(x_1, y_1)$  und  $(x_2, y_2)$  die Zentrumskoordinaten der beiden Detektionen darstellen.

## 6.5.2 Automatische Fehlerkorrektur

### Korrekturmuster und Heuristiken

Basierend auf der Analyse häufiger OCR-Fehler in der Gleisplanerkennung wurden regelbasierte Korrekturmuster entwickelt, die systematisch auftretende Fehler automatisch beheben können. Diese Korrekturmuster adressieren typische Fehlerquellen, die aus den Limitierungen der OCR-Engines sowie aus der spezifischen Natur von Gleisplandokumenten resultieren.

Ein häufiger Fehler betrifft die Groß- und Kleinschreibung bei alphanumerischen Codes. OCR-Engines tendieren dazu, Großbuchstaben als Kleinbuchstaben zu interpretieren, insbesondere bei geringer Schriftgröße oder suboptimaler Bildqualität. Die Korrektur erfolgt durch systematische Umwandlung aller Kleinbuchstaben in Großbuchstaben für Signalbezeichnungen und ähnliche Codes, da die standardisierte Notation ausschließlich Großbuchstaben vorsieht. Diese Transformation wird mit einer Konfidenz von 0.95 bewertet, da die Regel nahezu ausnahmslos anwendbar ist.

Leerzeichenfehler (Whitespace-Fehler) entstehen, wenn OCR-Engines fälschlicherweise Leerzeichen in zusammenhängende Zeichenketten einfügen. Dies tritt besonders bei Signalbezeichnungen auf, wo beispielsweise A101 als A 101 erkannt werden kann. Die Korrektur besteht in der Entfernung aller Leerzeichen aus alphanumerischen Codes. Diese Operation wird mit einer Konfidenz von 0.80 bewertet, da in seltenen Fällen legitime Leerzeichen existieren können, die jedoch im Kontext der Gleisplanerkennung unüblich sind.

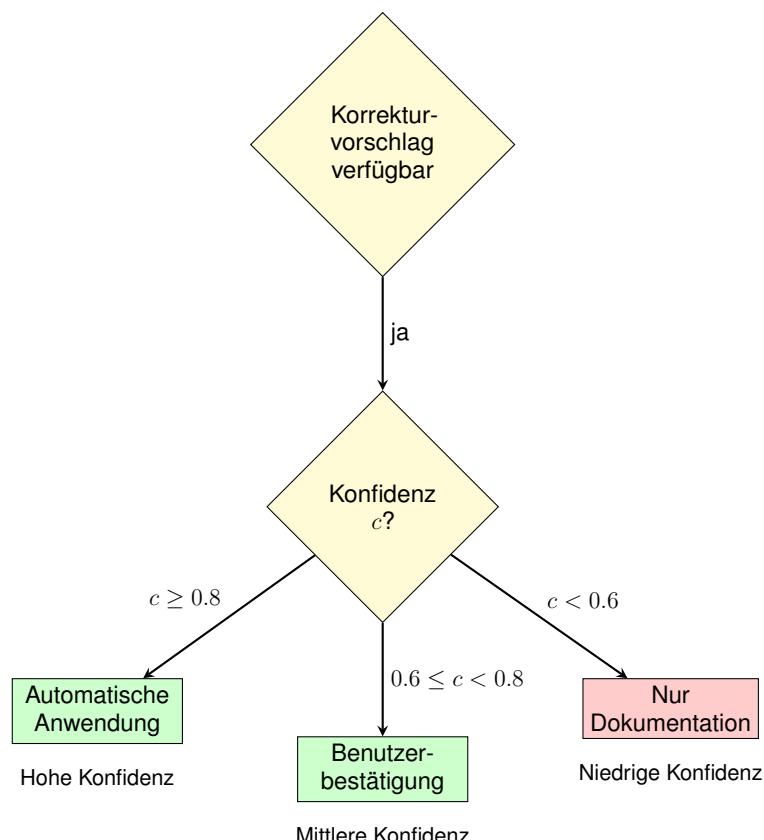
Die Normalisierung der Fahrtrichtungsangaben adressiert das Problem, dass OCR-Engines

die vorgeschriebenen Großbuchstaben A und B gelegentlich als Kleinbuchstaben a und b interpretieren. Die Korrektur erfolgt durch Konversion zu Großbuchstaben, sofern der erkannte Text genau einem Zeichen entspricht und entweder a oder b ist. Diese hochspezifische Regel erreicht eine Konfidenz von 0.90.

Für GKS-Nummern wurde beobachtet, dass OCR-Engines gelegentlich Trennzeichen wie Bindestriche oder Punkte in reine Ziffernfolgen einfügen. Die Korrekturstrategie entfernt alle nicht-numerischen Zeichen aus GKS Namen und validiert anschließend, ob das Ergebnis dem erwarteten Format von drei bis vier Ziffern entspricht. Aufgrund der höheren Unsicherheit dieser Transformation wird eine Konfidenz von 0.70 vergeben, da in Einzelfällen die entfernten Zeichen tatsächlich Teil der korrekten Bezeichnung sein könnten.

### Konfidenzbasierte Anwendungsstrategie

Die Anwendung automatischer Korrekturen erfolgt nicht undifferenziert, sondern wird durch ein konfidenzbasiertes Bewertungssystem gesteuert. Jede potenzielle Korrektur wird mit einem normalisierten Konfidenzwert im Intervall  $[0, 1]$  versehen, der die Zuverlässigkeit der Transformation quantifiziert. Dieser Wert wird aus mehreren Faktoren abgeleitet, insbesondere der Eindeutigkeit der Korrektur, der Häufigkeit des identifizierten Fehlermusters und dem Risiko einer Fehlkorrektur.



**Abbildung 6.22:** Konfidenzbasierte Entscheidungslogik für Korrekturvorschläge

Die Konfidenzwerte werden in drei Klassen kategorisiert, die unterschiedliche Handlungsempfehlungen implizieren. Korrekturen mit hoher Konfidenz, definiert als  $c \geq 0.8$ , repräsentieren eindeutige Transformationen mit minimalem Risiko. Diese beinhalten typischerweise regelbasierte Transformationen wie die Konversion von Klein- zu Großbuchstaben oder die Entfernung offensichtlich fehlerhafter Leerzeichen. Solche Korrekturen können mit hoher Sicherheit automatisch angewendet werden.

Korrekturen mit mittlerer Konfidenz im Bereich  $0.6 \leq c < 0.8$  indizieren plausible Transformationen, die jedoch ein gewisses Restrisiko bergen. Diese Kategorie umfasst Korrekturen, bei denen kontextabhängige Faktoren eine Rolle spielen oder wo die OCR-Ausgabe prinzipiell mehrdeutig sein könnte. Solche Vorschläge werden dem Benutzer präsentiert, aber nicht automatisch angewendet, sondern bedürfen einer manuellen Bestätigung.

Korrekturen mit niedriger Konfidenz, charakterisiert durch  $c < 0.6$ , werden als unsicher eingestuft und primär zu Dokumentationszwecken erfasst. Diese Vorschläge dienen dazu, den Benutzer auf potenzielle Probleme aufmerksam zu machen, ohne eine konkrete Handlungsempfehlung zu implizieren.

Die technische Umsetzung der Korrekturanwendung erfolgt durch selektive Modifikation des zugrundeliegenden Datencontainers. Für jedes identifizierte Problem mit dem Attribut automatisch korrigierbar und vorhandenem Korrekturvorschlag wird der entsprechende Eintrag im Datensatz lokalisiert und der Feldwert durch den vorgeschlagenen Wert ersetzt. Diese Operation erfolgt atomar für jede Korrektur und wird in einem Korrekturprotokoll dokumentiert, das die ursprünglichen Werte, die neuen Werte sowie die Zeilenummern der betroffenen Einträge enthält.

### 6.5.3 Integration in die Benutzeroberfläche

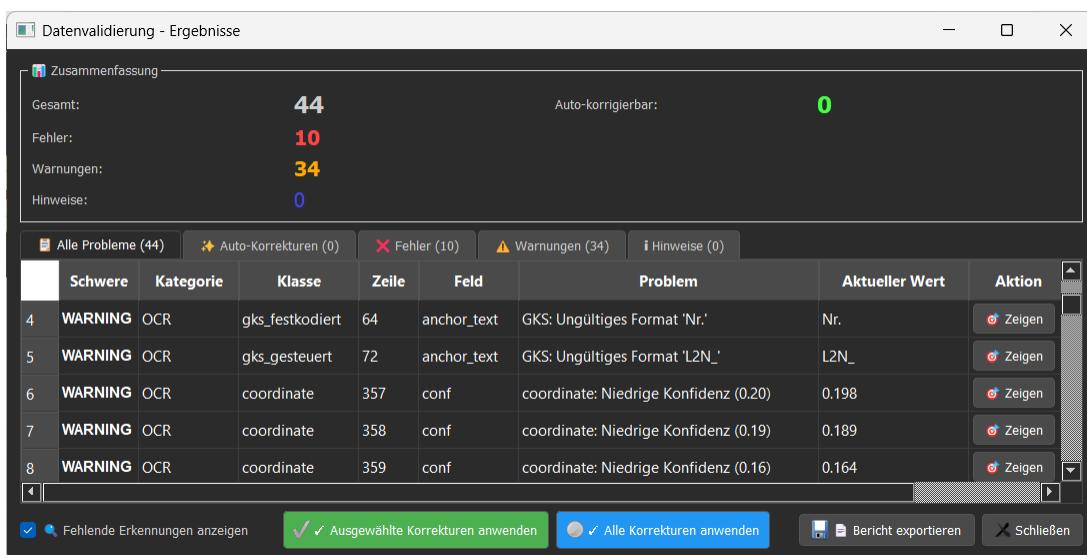
#### Dialogbasierte Validierungsansicht

Die Präsentation der Validierungsergebnisse erfolgt über einen modalen Dialog, der eine umfassende Übersicht über alle identifizierten Probleme sowie interaktive Funktionen zur Korrekturverwaltung bietet. Der Dialog ist als Tab-basiertes Interface konzipiert, das eine kategoriebasierte Navigation durch die Validierungsergebnisse ermöglicht und unterschiedliche Ansichten für verschiedene Arbeitsschritte bereitstellt.

Der erste Tab präsentiert die vollständige Liste aller identifizierten Validierungsprobleme in tabellarischer Form. Diese Gesamtansicht ermöglicht einen schnellen Überblick über den Validierungsstatus des gesamten Datensatzes und unterstützt die Priorisierung von Korrekturmaßnahmen. Die Tabelle umfasst Spalten für den Schweregrad, die Problemkategorie, die betroffene Objektklasse, die Zeilenidentifikation, das fehlerhafte Feld, eine Problembeschreibung sowie den aktuellen fehlerhaften Wert.

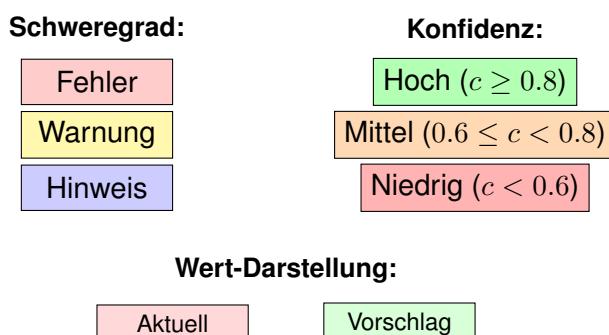
Ein spezialisierter Tab ist den automatisch korrigierbaren Problemen gewidmet und implemen-

tier ein interaktives Auswahlsystem. Jede Zeile in dieser Ansicht repräsentiert einen konkreten Korrekturvorschlag und ist mit einem Auswahlfeld versehen, das dem Benutzer die selektive Auswahl einzelner Korrekturen ermöglicht. Die visuelle Gestaltung unterstützt die Entscheidungsfindung durch eine farbliche Differenzierung: Der aktuelle fehlerhafte Wert wird mit rotem Hintergrund dargestellt, während der vorgeschlagene Korrekturwert grün hinterlegt ist. Die Konfidenz der Korrektur wird numerisch sowie durch Farbcodierung visualisiert, wobei grüne Schrift hohe Konfidenz, orange Schrift mittlere Konfidenz und rote Schrift niedrige Konfidenz signalisiert.



**Abbildung 6.23:** Schematische Darstellung der fünfteiligen Tab-Struktur des Validierungsdialogs

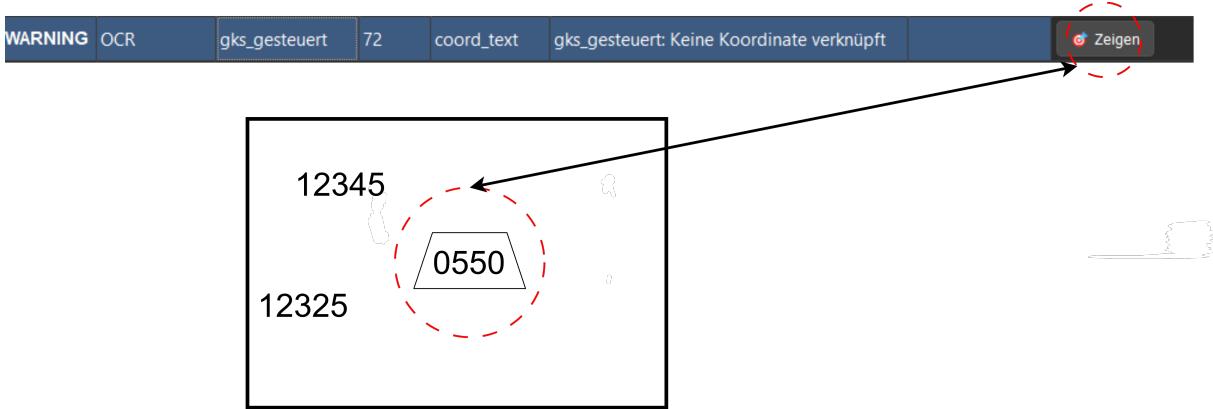
Weitere Tabs bieten gefilterte Ansichten, die ausschließlich Fehler, Warnungen oder Informationshinweise anzeigen. Diese kategoriespezifischen Ansichten reduzieren die kognitive Last bei der Fehleranalyse, indem sie eine fokussierte Bearbeitung nach Priorität ermöglichen. Kritische Fehler, die zwingend einer Korrektur bedürfen, können somit isoliert betrachtet werden, ohne durch weniger dringende Warnungen oder informative Hinweise überlagert zu werden.



**Abbildung 6.24:** Farbkodierung für Validierungsprobleme und Korrekturvorschläge

### Jump-to-Detection Funktionalität

Eine wesentliche Herausforderung bei der manuellen Validierung besteht in der effizienten Lokalisierung problematischer Erkennungen im Originaldokument. Das System adressiert dies durch eine integrierte Navigation, die eine direkte Verlinkung zwischen Validierungsproblemen und der entsprechenden Position im PDF-Viewer ermöglicht.



**Abbildung 6.25:** Jump-to-Detection Workflow zwischen Validierungsdialog und Gleisplan-Ansicht

Für Validierungsprobleme, die eine räumliche Position besitzen, wird in der Tabelle eine zusätzliche Aktionsspalte mit einem Navigationsbutton bereitgestellt. Die Aktivierung dieses Buttons löst ein Signal aus, das die Identifikation der betroffenen Zeile sowie die räumlichen Koordinaten ( $x, y$ ) der Erkennung transportiert. Dieses Signal wird vom übergeordneten Workspace-Modul empfangen und verarbeitet. Die Verarbeitung umfasst mehrere Schritte: Zunächst wird die entsprechende Zeile im Baumansicht-Widget des Workspace selektiert, wodurch die Detaillinformationen der Erkennung im unteren Panel angezeigt werden. Anschließend wird die PDF-Ansicht auf die entsprechende Seite navigiert und die Ansicht so zentriert, dass die Position der Erkennung im sichtbaren Bereich liegt. Abschließend wird ein visuelles Highlighting der Erkennungsregion aktiviert, typischerweise durch Einblendung eines farbigen Rahmens oder einer semitransparenten Überlagerung.

Die technische Implementierung dieser Funktionalität basiert auf dem Signal-Slot-Mechanismus von PyQt5. Der Validierungsdialog emittiert ein Signal, das die Zeilenidentifikation sowie die Position als Tupel überträgt. Der Workspace registriert einen Slot für dieses Signal, der die beschriebene Navigations- und Highlighting-Logik implementiert.

### Interaktiver Validierungsworkflow

Der typische Arbeitsablauf bei der Validierung und Korrektur extrahierter Daten gliedert sich in mehrere Phasen. Zunächst wählt der Benutzer die Validierungsfunktion im Anwendungsmenü aus, wodurch alle konfigurierten Validierungsprüfungen für den aktuellen Datensatz gestartet werden. Die Validierungsengine analysiert sequenziell alle Daten und sammelt identifizierte

Probleme in einer strukturierten Liste.

Nach Abschluss der Validierung wird der Ergebnisdialog präsentiert, der eine statistische Zusammenfassung der identifizierten Probleme am oberen Rand anzeigt. Diese Zusammenfassung quantifiziert die Anzahl der Fehler, Warnungen und Hinweise sowie die Anzahl automatisch korrigierbarer Probleme, aufgeschlüsselt nach Konfidenzklassen. Diese Übersicht ermöglicht eine schnelle Einschätzung des Validierungsstatus und der erforderlichen Korrekturmaßnahmen.

Der Benutzer navigiert anschließend durch die verschiedenen Tabs, um die Detailansichten der Probleme zu inspizieren. Im Tab für automatische Korrekturen werden die Korrekturvorschläge geprüft und durch Aktivierung oder Deaktivierung der zugehörigen Checkboxen selektiert. Die Anzahl der selektierten Korrekturen wird dynamisch aktualisiert und in der Beschriftung des Anwendungs-Buttons reflektiert.

Die Anwendung der selektierten Korrekturen erfolgt durch Aktivierung des entsprechenden Buttons, was zunächst einen Bestätigungsdialog aufruft. Dieser Dialog präsentiert eine Zusammenfassung der anzuwendenden Änderungen und warnt den Benutzer, dass die Modifikationen sofort im Datensatz wirksam werden. Nach Bestätigung werden die Korrekturen sequenziell auf den Datensatz angewendet, und der Dialog schließt sich. Die korrigierten Daten sind unmittelbar im Workspace sichtbar und können anschließend persistent in der Datenbank gespeichert werden.

#### 6.5.4 Persistierung und Export der Validierungsergebnisse

##### Export-Formate für Validierungsberichte

Das System bietet zwei komplementäre Export-Optionen für Validierungsergebnisse, die unterschiedliche Anwendungsfälle adressieren. Der detaillierte Textbericht stellt eine menschenlesbare, narrative Darstellung der Validierungsergebnisse bereit, während der tabellarische CSV-Export eine maschinenlesbare Repräsentation für statistische Auswertungen und Weiterverarbeitung ermöglicht.

Der Textbericht beginnt mit einer strukturierten Kopfzeile, die den Berichtstyp und den Generierungszeitpunkt dokumentiert. Es folgt eine statistische Zusammenfassung, die alle aggregierten Kennzahlen der Validierung präsentiert, einschließlich der Gesamtanzahl identifizierter Probleme, der Verteilung nach Schweregraden sowie der Anzahl automatisch korrigierbarer Probleme differenziert nach Konfidenzklassen. Falls automatische Korrekturen bereits angewendet wurden, werden diese in einem separaten Abschnitt aufgelistet, wobei für jede Korrektur die Zeilenidentifikation, das betroffene Feld sowie die Transformation vom ursprünglichen zum korrigierten Wert dokumentiert wird.

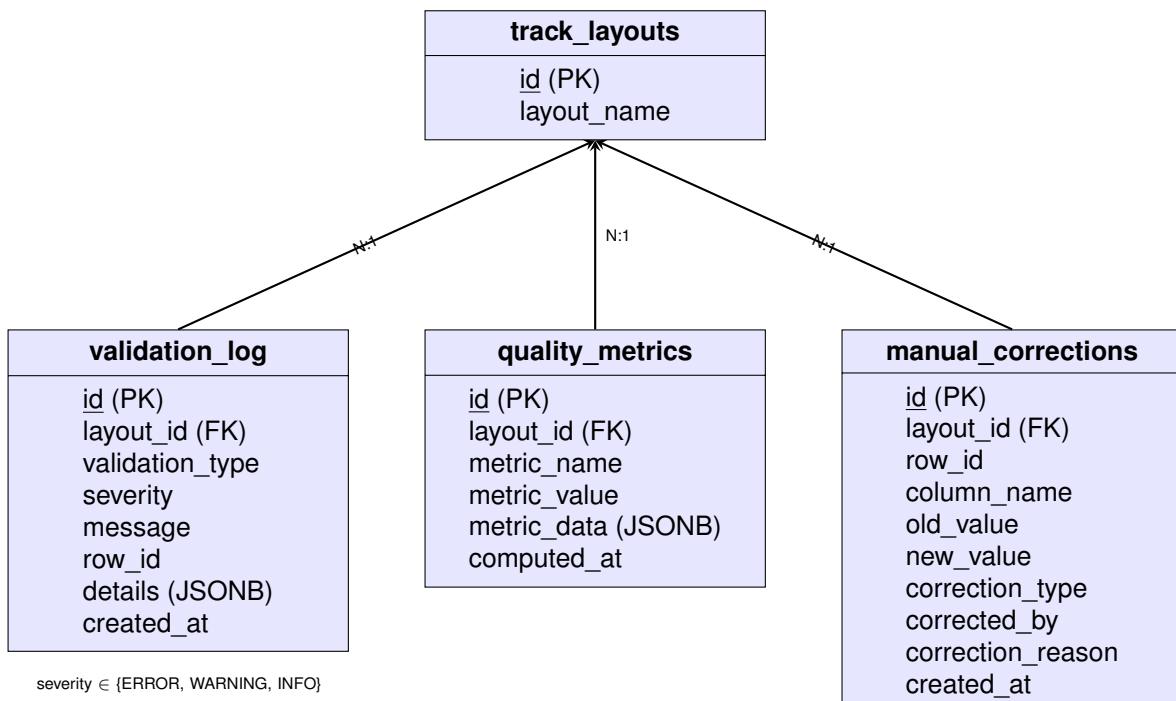
Der Hauptteil des Textberichts gruppiert die identifizierten Probleme nach Kategorien. Für jede Kategorie wird ein dedizierter Abschnitt erstellt, der alle zugehörigen Probleme in strukturierter Form auflistet. Jedes Problem wird durch mehrere Zeilen beschrieben: die Zeilenidentifikation

mit Schweregrad, das betroffene Feld, die detaillierte Problembeschreibung, den aktuellen fehlerhaften Wert sowie gegebenenfalls den Korrekturvorschlag inklusive Konfidenzwert. Diese narrative Struktur ermöglicht eine intuitive Erfassung der Validierungsergebnisse und eignet sich für manuelle Überprüfungen oder als Dokumentation des Validierungsprozesses.

Der tabellarische CSV-Export repräsentiert die Validierungsergebnisse in einer flachen, relationalen Struktur. Jede Zeile der CSV-Datei korrespondiert zu einem identifizierten Problem, wobei folgende Spalten definiert sind: Schweregrad als kategoriale Variable, Kategorie zur Klassifikation des Problemtyps, Zeilenidentifikation zur Referenzierung im ursprünglichen Datensatz, betroffenes Feld als Spaltenname, detaillierte Problembeschreibung als Freitext, aktueller fehlerhafter Wert, Korrekturvorschlag falls vorhanden, und Konfidenzwert des Vorschlags. Diese Struktur ermöglicht den Import in Tabellenkalkulationssoftware oder statistische Analysewerkzeuge und unterstützt quantitative Auswertungen der Validierungsergebnisse.

### Datenbankbasierte Persistierung

Zur langfristigen Speicherung und Nachverfolgbarkeit werden Validierungsergebnisse in einer PostgreSQL-Datenbank persistiert. Das Datenbankschema (Abbildung 6.26) umfasst drei spezialisierte Tabellen für Validierungsprotokolle, Qualitätsmetriken und manuelle Korrekturen.



**Abbildung 6.26:** PostgreSQL Datenbankschema für Validierungspersistierung

Die Tabelle für Validierungsprotokolle (`validation_log`) speichert detaillierte Informationen zu jedem identifizierten Problem. Die Struktur umfasst eine eindeutige Identifikation des Protokoleintrags, eine Referenz zum zugehörigen Gleisplan, die Klassifikation des Validierungstyps, den Schweregrad als standardisierte Kategorie (ERROR, WARNING, INFO), eine textuelle

Problembeschreibung, eine optionale Referenz zur betroffenen Zeile im extrahierten Datensatz, ein JSON-Feld für zusätzliche kontextuelle Informationen sowie einen Zeitstempel der Protokollierung. Diese Struktur ermöglicht sowohl die Rekonstruktion des Validierungszustands zu einem bestimmten Zeitpunkt als auch die Analyse von Fehlertrends über mehrere Extraktionsdurchläufe hinweg.

Die Qualitätsmetriken-Tabelle (quality)aggregiert quantitative Kennzahlen zur Bewertung der Extraktionsqualität. Gespeichert werden der Name der Metrik als Identifikator, ein numerischer Wert für quantitative Metriken, ein JSON-Feld für strukturierte Metrikdaten sowie ein Zeitstempel der Berechnung. Typische Metriken umfassen die durchschnittliche YOLO-Konfidenz aggregiert über alle Klassen sowie klassenspezifisch, die OCR-Erfolgsrate für Koordinaten und Signalbezeichnungen als Quotient erfolgreich erkannter Texte zur Gesamtzahl der Objekte, die Linking-Erfolgsrate als Anteil der Objekte mit korrekt verknüpften Koordinaten, sowie die Anzahl niedrig-konfiderter Erkennungen differenziert nach Objektklasse. Diese Metriken ermöglichen eine quantitative Bewertung der Systemleistung und unterstützen die Identifikation von Optimierungspotenzialen.

Die dritte Tabelle (manual\_corrections)protokolliert manuelle Korrekturen, die durch den Benutzer nach Abschluss der automatischen Validierung vorgenommen werden. Für jede Korrektur werden die Zeilenidentifikation, der Name der modifizierten Spalte, der ursprüngliche Wert vor Korrektur, der neue Wert nach Korrektur, eine Typisierung der Korrektur zur Kategorisierung, die Identifikation des korrigierenden Benutzers, eine optionale textuelle Begründung sowie ein Zeitstempel gespeichert. Diese Daten dienen mehreren Zwecken: Sie ermöglichen eine vollständige Nachvollziehbarkeit aller Datenmodifikationen, unterstützen die Analyse häufiger Korrekturmuster zur Identifikation systematischer Fehler in der Extraktionspipeline, und liefern wertvolle Trainingsdaten für die iterative Verbesserung der zugrundeliegenden Modelle.

Zur Unterstützung statistischer Analysen wurde eine Datenbankansicht implementiert, die Korrekturstatistiken aggregiert. Diese Ansicht fasst Korrekturen nach Gleisplan und Spaltenname zusammen und berechnet die Gesamtanzahl der Korrekturen, die Anzahl betroffener unterschiedlicher Zeilen sowie den Zeitraum der ersten und letzten Korrektur. Diese aggregierten Daten ermöglichen die Identifikation besonders fehleranfälliger Datenfelder und unterstützen die Priorisierung von Verbesserungsmaßnahmen.

Die Implementierung der Datenbankpersistierung nutzt prepared statements und parametrisierte Abfragen zur Vermeidung von SQL-Injection-Vulnerabilitäten. Transktionale Semantik gewährleistet die atomare Speicherung zusammengehöriger Datensätze. Indizes auf häufig abgefragten Spalten optimieren die Leistung bei Analyseabfragen über große Validierungsdatenmengen.

### 6.5.5 Zusammenfassung

Das Validierungssystem kombiniert regelbasierte Prüfungen mit intelligenten Korrekturvorschlägen und bietet eine benutzerfreundliche UI zur Qualitätssicherung. Durch Persistierung der

Ergebnisse in PostgreSQL werden langfristige Qualitätsanalysen und iterative Modellverbesserungen ermöglicht. Korrekturen mit niedriger Konfidenz oder mehrdeutigen Fällen werden dem Benutzer zur manuellen Prüfung vorgelegt.

## 6.6 Unterstützende Komponenten

Neben der Kernfunktionalität der Extraktionspipeline (Objekterkennung, Texterkennung, Symbol-Text-Verknüpfung) wurden mehrere unterstützende Komponenten entwickelt, die für den praktischen Einsatz des Systems in realen Anwendungsszenarien erforderlich sind. Diese Komponenten adressieren die Anforderungen an Versionsverwaltung, Rückverfolgbarkeit und persistente Datenhaltung, welche für den operativen Einsatz in ingenieurwissenschaftlichen Workflows unerlässlich sind.

### 6.6.1 Versionsvergleich und Änderungsdetektion

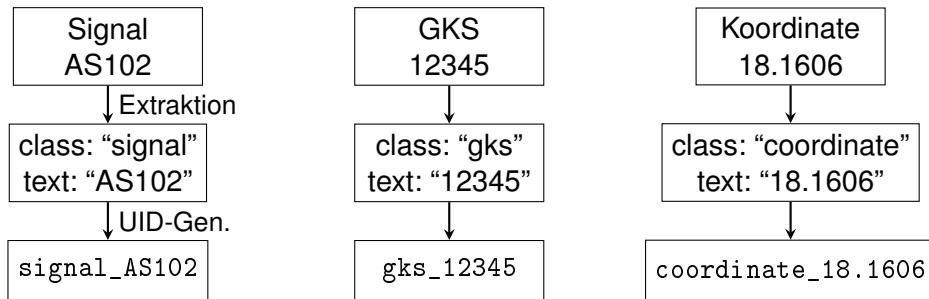
Die systematische Identifikation und Dokumentation von Änderungen zwischen verschiedenen Versionen eines Gleisplans stellt eine zentrale Anforderung in der Eisenbahnplanung dar. Diese Funktionalität wird vom Benutzer manuell über die Benutzeroberfläche initiiert, nicht als Teil der automatischen Verarbeitungspipeline. Zur Realisierung dieser Funktionalität wurde ein Diff-Algorithmus implementiert, der auf der Generierung eindeutiger Identifikatoren und mengentheoretischen Operationen basiert.

#### Generierung eindeutiger Identifikatoren

Für die eindeutige Identifikation extrahierter Objekte über verschiedene Planversionen hinweg wird ein deterministischer Unique Identifier (UID) generiert. Die UID-Generierung erfolgt durch Konkatenation der Objektklasse, des normalisierten Textinhalts und eines räumlichen Index, um Dopplungen bei identischem Textinhalt zu vermeiden:

$$\text{UID}(o) = \text{class}(o) \oplus \text{normalize}(\text{content}(o)) \oplus \text{GridIndex}(x, y) \quad (6.33)$$

wobei  $\text{GridIndex}(x,y)$  die Koordinaten auf ein grobes Raster abbildet, um Robustheit gegen minimale Pixelverschiebungen zu gewährleisten, während Identität bei gleicher Semantik an verschiedenen Orten unterschieden wird.  $\oplus$  bezeichnet die String-Konkatenation und die Normalisierungsfunktion  $\text{normalize} : \Sigma^* \rightarrow \Sigma^*$  Leerzeichen entfernt und Sonderzeichen standardisiert, um konsistente Identifikatoren zu gewährleisten.



**Abbildung 6.27:** Schematische Darstellung des UID-Generierungsprozesses für verschiedene Objekttypen

### Beispiele für generierte UIDs:

- Signal mit Bezeichnung “AS102” → signal\_AS102
- GKS mit Kennung “12345” → gks\_12345
- Kilometerangabe “18.1606” → coordinate\_18.1606

Die Verwendung von UIDs ermöglicht eine effiziente Objektzuordnung zwischen Planversionen, selbst bei Änderungen der räumlichen Position oder anderer Attribute.

### Differenzalgorithmus

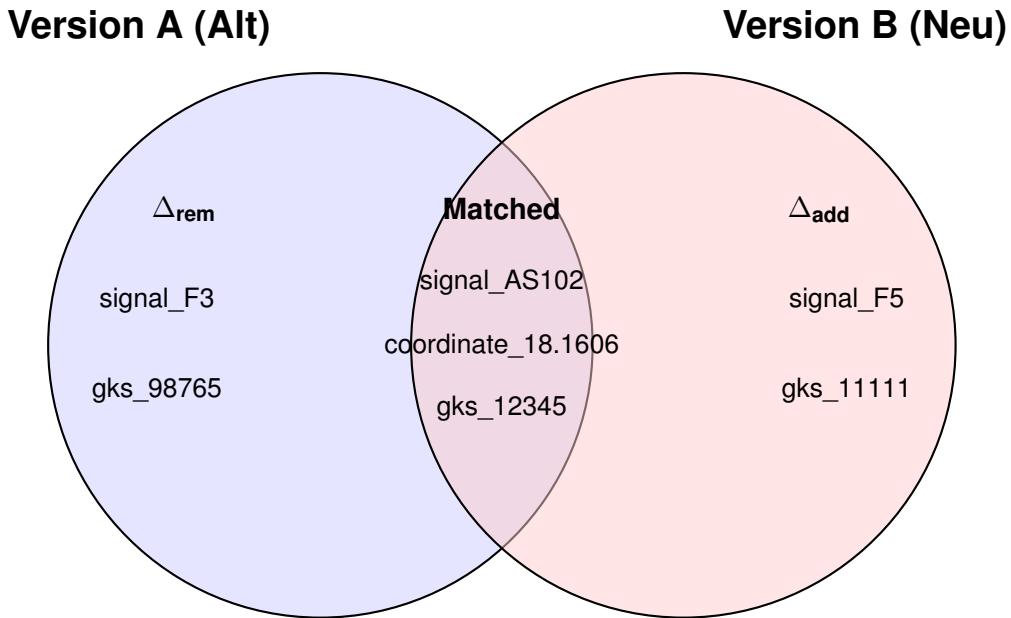
Der Vergleichsalgorithmus verwendet den *Hungarian Algorithm* (Kuhn-Munkres-Algorithmus) zur optimalen Zuordnung von Elementen zwischen zwei Extraktionsmengen  $A$  (Altversion) und  $B$  (Neuversion). Im Gegensatz zu einfachen mengentheoretischen Operationen ermöglicht dieser Ansatz eine robuste Behandlung von Duplikaten und Elementen ohne eindeutige Identifikatoren.

### Resultierende Änderungsmengen:

$$\Delta_{\text{add}} = \{b \in B \mid \nexists a \in A : \text{matched}(a, b)\} \quad (\text{hinzugefügte Objekte}) \quad (6.34)$$

$$\Delta_{\text{rem}} = \{a \in A \mid \nexists b \in B : \text{matched}(a, b)\} \quad (\text{entfernte Objekte}) \quad (6.35)$$

$$\Delta_{\text{mov}} = \{(a, b) \mid \text{matched}(a, b) \wedge \text{coord}(a) \neq \text{coord}(b)\} \quad (\text{verschobene Objekte}) \quad (6.36)$$



**Abbildung 6.28:** Mengentheoretische Visualisierung des Diff-Algorithmus

Für erfolgreich zugeordnete Objektpaare (matched) wird geprüft, ob sich der Kilometrierungswert (coord\_value) geändert hat. Ein Objekt gilt als verschoben, falls:

$$\text{isMoved}(o_A, o_B) = \begin{cases} \text{true,} & \text{falls } |\text{coord}(o_A) - \text{coord}(o_B)| > 0 \\ \text{false,} & \text{sonst} \end{cases} \quad (6.37)$$

Die Schwere der Verschiebung wird nach der Differenz in Metern klassifiziert:

$$\text{severity}(\Delta_{\text{coord}}) = \begin{cases} \text{MINOR,} & \text{falls } \Delta_{\text{coord}} < 5 \text{ m} \\ \text{MODERATE,} & \text{falls } 5 \text{ m} \leq \Delta_{\text{coord}} \leq 20 \text{ m} \\ \text{MAJOR,} & \text{falls } \Delta_{\text{coord}} > 20 \text{ m} \end{cases} \quad (6.38)$$

### Strukturierte Änderungsdokumentation

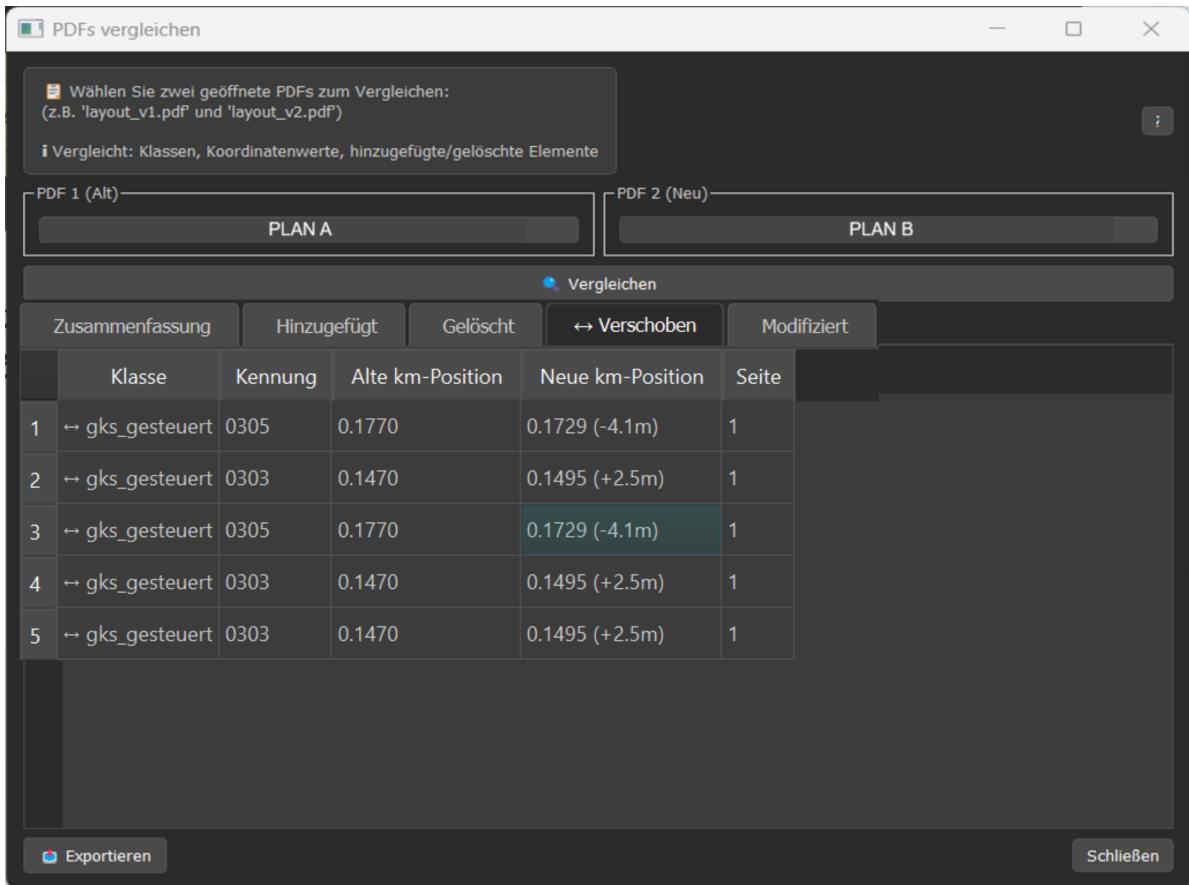
Die identifizierten Änderungen werden in einem strukturierten Change Report dokumentiert, der als Excel-Datei exportiert wird und folgende Informationen je Änderung enthält:

UID	Änderungstyp	Zustand (Alt)	Zustand (Neu)
signal_AS102	Verschoben	km 18.1606	km 18.1650 (+4,4m)
signal_F5	Hinzugefügt	–	km 23.4500
gks_98765	Entfernt	km 15.2300	–

**Tabelle 6.13:** Beispielhafte Change Report Struktur mit verschiedenen Änderungstypen

Diese strukturierte Dokumentation ermöglicht eine effiziente Nachvollziehbarkeit von Planänderungen und unterstützt Quality-Assurance-Prozesse in der Eisenbahnplanung. Die Implementie-

rung umfasst eine dedizierte grafische Benutzeroberfläche zur Visualisierung und Analyse der Änderungen, wie in Abbildung 6.29 dargestellt.



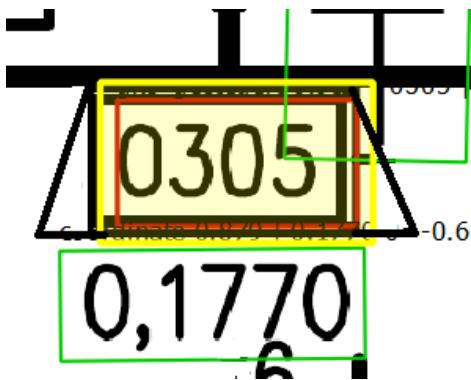
**Abbildung 6.29:** Versionsvergleichs-Dialog mit tabellarischer Auflistung aller Änderungen zwischen zwei Planversionen

Der Dialog zeigt eine tabellarische Übersicht aller identifizierten Änderungen mit Spalten für Objektklasse, Kennung, alte und neue km-Position sowie Seitenzuordnung. In diesem Beispiel wurden fünf GKS-Koordinaten als verschoben erkannt. Die Änderungstypen werden durch Icons visualisiert: Hinzugefügte Elemente erhalten ein grünes Plus-Symbol, entfernte Elemente ein rotes Minus-Symbol, und verschobene Elemente werden gelb hervorgehoben.

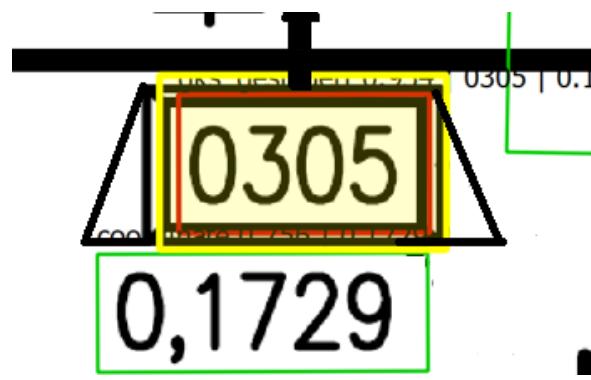
**Anmerkung zu mehrfachen Einträgen:** Die Tabelle zeigt drei Einträge für GKS 0303 und zwei Einträge für GKS 0305. Dies ist kein Fehler, sondern resultiert aus der Struktur der verglichenen Gleispläne: Diese enthalten drei Streckenabschnitte (Varianten), wobei dieselbe GKS-Kennung in mehreren Abschnitten vorkommen kann. Jede Instanz wird separat verglichen, da sie unterschiedliche Kilometrierungen aufweisen kann.

Diese Darstellung erfüllt Anforderung FA-011 und ermöglicht Ingenieuren eine schnelle Identifikation relevanter Planänderungen ohne manuelle Vergleichsarbeit.

Ein konkretes Beispiel einer detektierten Koordinatenänderung zeigen die Abbildungen 6.30 und 6.31 für die GKS mit der Kennung 0305.



**Abbildung 6.30:** GKS 0305 in Version 1:  
Koordinate 0.1770

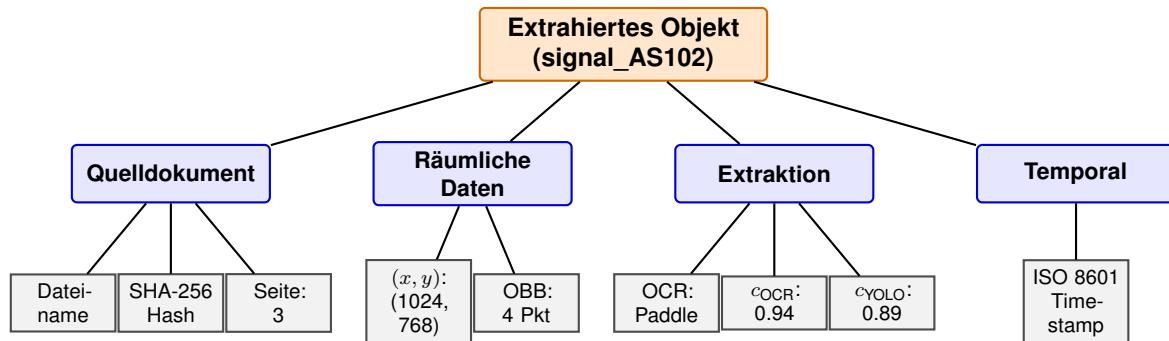


**Abbildung 6.31:** GKS 0305 in Version 2:  
Koordinate 0.1729

Die automatische Änderungsdetektion identifizierte eine Verschiebung um  $\Delta = -0,0041$  km (ca. 4,1 Meter) durch UID-basierten Objektvergleich. Der Algorithmus erkannte, dass beide Versionen dasselbe Objekt (gks\_gesteuert\_0305) beschreiben, jedoch unterschiedliche Koordinatenwerte aufweisen. Solche Präzisionsänderungen im Meterbereich sind in der Feinplanung von Bahnanlagen kritisch und müssen dokumentiert werden, da sie die Positionierung sicherheitsrelevanter Komponenten betreffen.

## 6.6.2 Rückverfolgbarkeit zur Quelldokumentation

Die vollständige Rückverfolgbarkeit extrahierter Daten zur Originalquelle ist essentiell für die Validierung und manuelle Nachkorrektur von Extraktionsergebnissen. Hierzu wird für jeden extrahierten Datensatz ein umfassender Metadatensatz persistiert.



**Abbildung 6.32:** Hierarchische Struktur der Metadaten für ein extrahiertes Objekt

## Metadatenmodell

Das Metadatenmodell umfasst folgende Informationsebenen:

### Quelldokument-Identifikation:

- Dateiname der Quell-PDF ( $f_{source}$ )
- Kryptographischer Hash (SHA-256) zur Integritätssicherung

- Seitennummer innerhalb des Dokuments ( $p \in \mathbb{N}$ )

### Räumliche Lokalisierung:

- Zentroid-Koordinaten in absoluten Pixelkoordinaten:  $(x, y) \in \mathbb{R}^2$
- Oriented Bounding Box Geometrie:  $\text{OBB} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
- Die OBB-Präsentation ermöglicht die präzise Visualisierung rotierter Textelemente

### Extraktionsdetails:

- Verwendete OCR-Engine ( $e \in \{\text{PaddleOCR}, \text{Tesseract}, \text{EasyOCR}\}$ )
- Konfidenzscore der Objekterkennung:  $c_{\text{YOLO}} \in [0, 1]$
- Konfidenzscore der Texterkennung:  $c_{\text{OCR}} \in [0, 1]$
- Angewandte Linking-Methode ( $m \in \{\text{standard}, \text{adaptive}\}$ )

### Temporale Zuordnung:

- ISO 8601 [106] formatierter Zeitstempel der Extraktion

### Integration in die Benutzeroberfläche

Die Metadaten werden zur Realisierung einer bidirektionalen Navigation zwischen Datenansicht und Quelldokument genutzt:

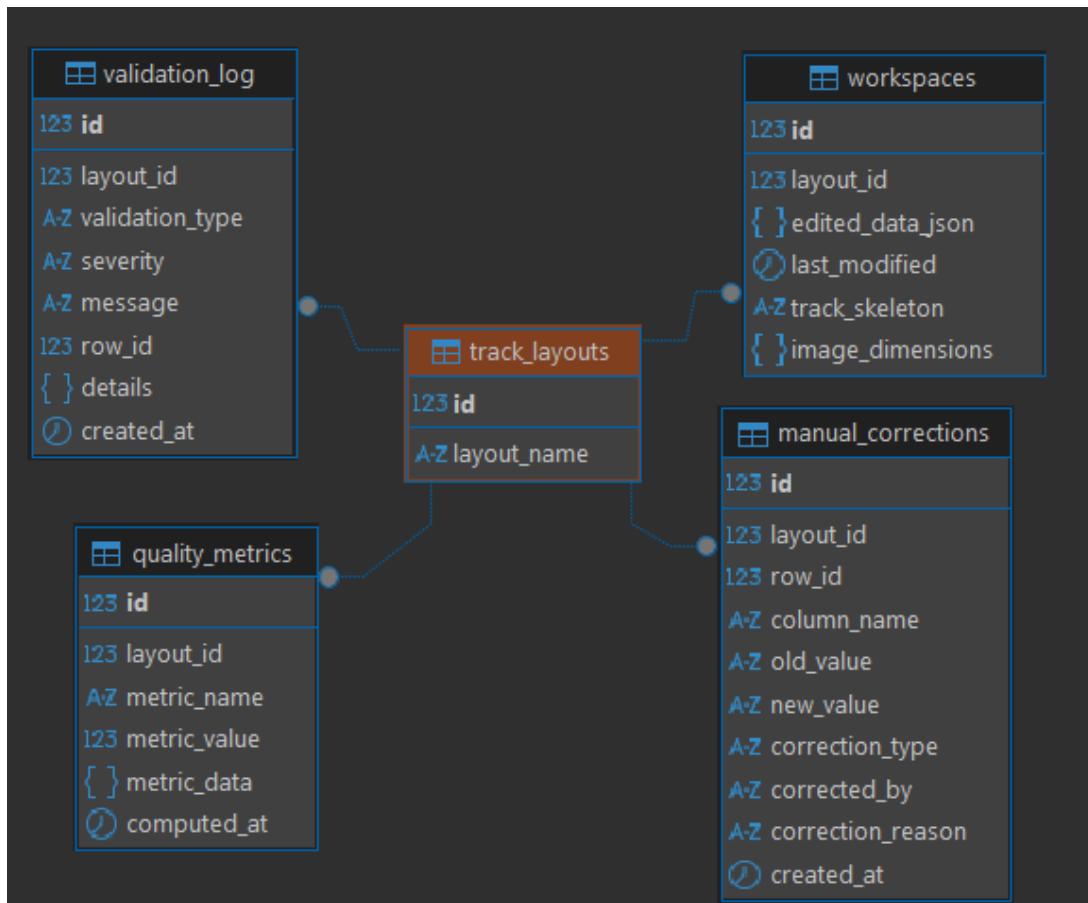
**Tabellenbasierte Navigation:** Beim Anklicken eines Datensatzes in der Ergebnistabelle wird der PDF-Viewer auf die entsprechende Seite navigiert, auf die dokumentierten Koordinaten zentriert und das Objekt mittels farbkodiertem Overlay hervorgehoben. Die OBB-Geometrie ermöglicht dabei eine pixelgenaue Umrandung auch bei rotierten Textelementen.

**PDF-basierte Navigation:** Interaktives Anklicken eines visualisierten Objekts im PDF-Viewer triggert ein automatisches Scrolling zur korrespondierenden Zeile in der Ergebnistabelle, wodurch die zugehörigen Extraktionsdaten unmittelbar einsehbar werden.

Diese Funktionalität reduziert den kognitiven Aufwand bei der manuellen Validierung signifikant und ermöglicht eine effiziente Identifikation systematischer Extraktionsfehler.

### 6.6.3 Persistente Datenhaltung

Das System verwendet PostgreSQL (Version 14.9) zur persistenten Speicherung von Extraktionsergebnissen, Workspace-Zuständen und Qualitätsmetriken.



**Abbildung 6.33:** Datenbankschema mit Kardinalitäten (vereinfachte Darstellung)

## Datenbankschema

Das Datenbankschema ist auf die Anforderungen eines dokumentbasierten Workflows optimiert und nutzt JSONB-Datentypen für flexible Speicherung semi-strukturierter Extraktionsdaten:

Relation: `track_layouts`

```

-----
id          SERIAL PRIMARY KEY
layout_name  TEXT UNIQUE NOT NULL
  
```

Relation: `workspaces`

```

-----
id          SERIAL PRIMARY KEY
layout_id    INTEGER UNIQUE NOT NULL REFERENCES track_layouts(id)
            ON DELETE CASCADE
edited_data_json  JSONB NOT NULL          -- Extraktionsergebnisse
track_skeleton    TEXT                   -- Komprimiertes Gleisskelett
image_dimensions  JSONB                  -- Seitenabmessungen
last_modified     TIMESTAMPTZ DEFAULT NOW()
  
```

```
Relation: validation_log
-----
id          SERIAL PRIMARY KEY
layout_id    INTEGER NOT NULL REFERENCES track_layouts(id)
validation_type VARCHAR(100) NOT NULL
severity      VARCHAR(20) CHECK (severity IN ('ERROR', 'WARNING', 'INFO'))
message       TEXT NOT NULL
row_id        INTEGER                      -- Referenz zu Datensatz
details       JSONB
created_at    TIMESTAMPTZ DEFAULT NOW()

Relation: quality_metrics
-----
id          SERIAL PRIMARY KEY
layout_id    INTEGER NOT NULL REFERENCES track_layouts(id)
metric_name  VARCHAR(100) NOT NULL
metric_value REAL
metric_data   JSONB
computed_at  TIMESTAMPTZ DEFAULT NOW()

Relation: manual_corrections
-----
id          SERIAL PRIMARY KEY
layout_id    INTEGER NOT NULL REFERENCES track_layouts(id)
row_id      INTEGER NOT NULL
column_name  VARCHAR(100) NOT NULL
old_value    TEXT
new_value    TEXT
correction_type VARCHAR(50)
corrected_by  VARCHAR(100) DEFAULT 'user'
correction_reason TEXT
created_at    TIMESTAMPTZ DEFAULT NOW()

-- Performance-Indizes
CREATE INDEX idx_validation_log_layout ON validation_log(layout_id);
CREATE INDEX idx_validation_log_severity ON validation_log(severity);
CREATE INDEX idx_quality_metrics_layout ON quality_metrics(layout_id);
CREATE INDEX idx_manual_corrections_layout ON manual_corrections(layout_id);
```

## JSONB-basierte Datenspeicherung

Die Extraktionsergebnisse werden als JSONB-Array in `edited_data_json` gespeichert. Diese Designentscheidung bietet mehrere Vorteile gegenüber einer vollständig normalisierten Tabellenstruktur:

**Datenstruktur:** Jeder Eintrag in `edited_data_json` repräsentiert ein extrahiertes Objekt mit folgender Struktur:

```
{
  "row_id": 42,
  "cls": "signal",
  "anchor_text": "AS102",
  "coord_text": "18.1606(GL2)",
  "coord_value": 18.1606,
  "x": 1024.5,
  "y": 768.3,
  "obb_points": [[x1,y1], [x2,y2], [x3,y3], [x4,y4]],
  "conf": 0.94,
  "fahrtrichtung": "A",
  ...
}
```

### Vorteile des JSONB-Ansatzes:

- **Schema-Flexibilität:** Neue Attribute können ohne Datenbankmigrationen hinzugefügt werden
- **Atomare Updates:** Gesamter Workspace wird als Einheit gespeichert, konsistent mit UI-Operationen
- **Effiziente Queries:** PostgreSQL JSONB unterstützt Indexierung und performante Abfragen
- **Versionierung:** Einfaches Speichern vollständiger Snapshots zu verschiedenen Zeitpunkten

## Verwendungszwecke

Die Datenbank erfüllt folgende funktionale Anforderungen im operativen Workflow:

- **Session-Persistierung:** Speicherung von Workspace-Zuständen ermöglicht Unterbrechung und spätere Fortsetzung der Arbeit
- **Validierungs-Tracking:** Die Tabelle `validation_log` dokumentiert alle identifizierten Datenqualitätsprobleme mit Schweregradklassifikation (ERROR, WARNING, INFO), was

eine systematische Fehleranalyse ermöglicht

- **Korrektur-Historisierung:** Manuelle Benutzerkorrekturen werden in `manual_corrections` protokolliert. Diese Daten können zur Identifikation systematischer Extraktionsfehler und zur Verbesserung der Pipeline genutzt werden
- **Qualitätsmetriken:** Aggregierte Qualitätskennzahlen (z.B. durchschnittliche OCR-Konfidenz, Vollständigkeitsraten) werden in `quality_metrics` persistiert für longitudinale Analysen
- **Export-Vorbereitung:** Vor dem Excel-Export werden Daten aus der Datenbank geladen, validiert und transformiert

## Datenbankzugriff

Die Datenbankinteraktion erfolgt über den PostgreSQL-Adapter `psycopg2` (Version 2.9.7). Die Datenbankoperationen sind in einem dedizierten Modul `database.py` gekapselt, welches folgende Kern-API bereitstellt:

### Workspace-Operationen:

- `save_workspace_data(name, data, skeleton, dims) → void`  
Persistiert Workspace mit Extraktionsdaten, optionalem Gleisskelett und Bildabmessungen
- `get_workspace_data(name) → (data, skeleton, dims)`  
Lädt gespeicherten Workspace inkl. Dekompression des Gleisskeletts

### Validierungs-Operationen:

- `save_validation_results(name, results) → void`  
Speichert Validierungsergebnisse mit automatischer Historisierung
- `get_validation_summary(name) → dict`  
Liefert Zusammenfassung nach Schweregrad aggregiert

### Korrektur-Operationen:

- `log_manual_correction(name, row, field, old, new) → void`  
Protokolliert Benutzerkorrektur für spätere Analyse
- `get_correction_statistics(name) → list`  
Analysiert Korrekturmuster zur Identifikation systematischer Fehler

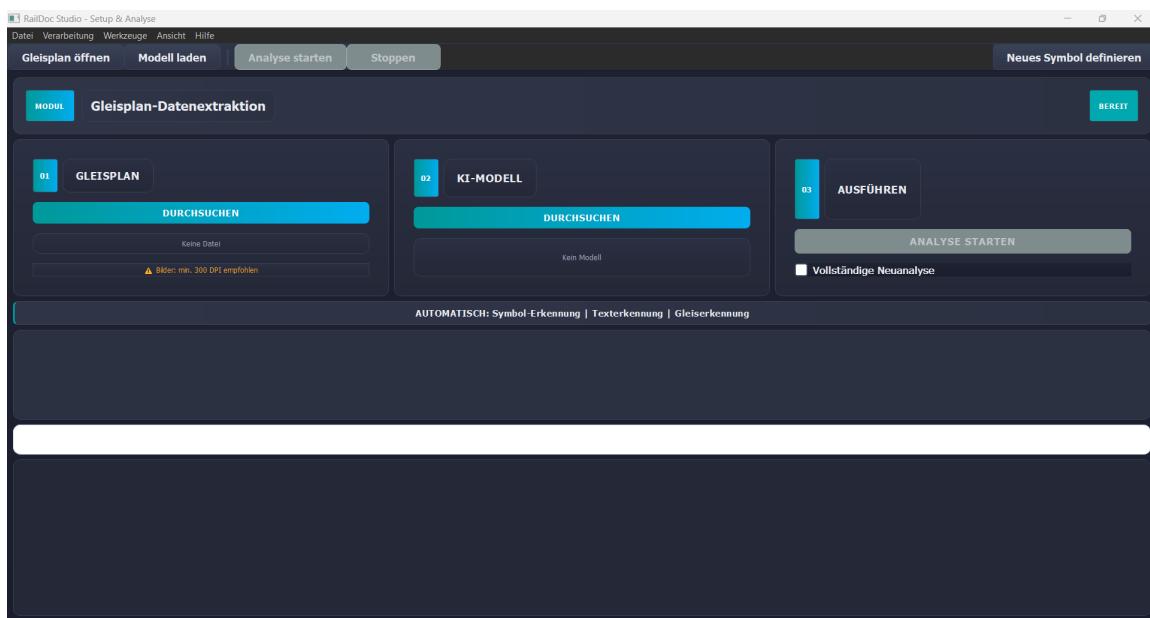
Die Verwendung von Connection-Pooling mittels Context Manager optimiert die Performance bei häufigen Datenbankzugriffen während des Extraktionsprozesses und stellt automatisches Rollback bei Fehlern sicher.

## 6.7 Benutzeroberfläche

Die Benutzeroberfläche dient als interaktive Komponente zur Qualitätskontrolle und manuellen Nachbearbeitung der extrahierten Daten. Sie adressiert die Anforderungen **FA-013** (GUI ohne CLI-Kenntnisse), **FA-012** (visuelle Validierung durch Bounding-Box-Overlays) und **NFA-001** (On-Premise-Verarbeitung als Desktop-Anwendung).

### 6.7.1 Setup- und Analysedialog

Der initiale Einstiegspunkt der Anwendung „RailDoc Studio“ ist der Setup- und Analysedialog (Abbildung 6.34), der einen geführten dreistufigen Workflow zur Initiierung der Datenextraktion implementiert.



**Abbildung 6.34:** Setup- und Analyse-Dialog der Anwendung „RailDoc Studio“

Der Workflow gliedert sich in drei sequenzielle Schritte:

1. **Gleisplan laden:** Auswahl der zu verarbeitenden PDF-Datei über einen Dateidialog. Das System akzeptiert PDF-Dateien beliebiger Größe und zeigt einen Hinweis zur empfohlenen Mindestauflösung von 300 DPI an, wobei 500 DPI für optimale Erkennungsleistung empfohlen werden (vgl. Anforderung **NFA-009**).
2. **KI-Modell laden:** Auswahl der trainierten YOLOv8-OBB Gewichtsdatei (best.pt). Diese explizite Trennung von Anwendung und Modell erfüllt Anforderung **NFA-008** (Update-Fähigkeit) und ermöglicht den Austausch von Modellversionen – beispielsweise nach einem Nachtraining mit zusätzlichen Symbolklassen – ohne Neukompilierung der Anwendung.
3. **Analyse starten:** Initiierung der automatischen Verarbeitungspipeline durch Betätigung der Schaltfläche „Analyse starten“. Der Fortschrittsbalken visualisiert die drei Pipeline-Stufen:

Symbol-Erkennung (YOLO), Texterkennung (OCR) und Gleiserkennung (Linking).

Die Checkbox „Vollständige Neuanalyse“ ermöglicht das Überschreiben bereits in der PostgreSQL-Datenbank gespeicherter Ergebnisse, was für iterative Verbesserungen oder nach Modell-Updates relevant ist. Die Schaltfläche „Neues Symbol definieren“ (rechts oben) öffnet einen Assistenten zur Definition zusätzlicher Symbolklassen und demonstriert die Erweiterbarkeit des Systems gemäß Anforderung **NFA-008**.

Nach Abschluss der automatischen Analyse wechselt die Anwendung automatisch in den Bearbeitungs- und Korrekturmodus (6.36), in dem die extrahierten Daten validiert und bei Bedarf manuell korrigiert werden können.

### 6.7.2 Architektur und Designentscheidungen

Die Architektur der Benutzeroberfläche basiert auf dem Model-View-Controller (MVC) Entwurfsmuster, wobei die Datenhaltung (Model), die Visualisierung (View) und die Ereignisverarbeitung (Controller) klar voneinander getrennt sind. Diese Architektur gewährleistet Wartbarkeit und ermöglicht die unabhängige Weiterentwicklung der einzelnen Schichten.

Die Hauptkomponenten gliedern sich in:

- **Hauptfenster:** Verwaltung mehrerer parallel geöffneter PDF-Dokumente in Tab-Ansicht
- **Arbeitsbereich:** Dedizierter Anzeigebereich für ein einzelnes PDF-Dokument mit allen zugehörigen Extraktionsergebnissen
- **Interaktive PDF-Ansicht:** Visualisierung des Originaldokuments mit überlagerten Erkennungsergebnissen
- **Hierarchische Ergebnistabelle:** Strukturierte Darstellung aller extrahierten Daten mit Excel-ähnlicher Editierfunktionalität
- **Validierungs-Dialog:** Spezialisierte Komponente zur systematischen Fehlerprüfung und -korrektur
- **Versionsvergleichs-Modul:** Werkzeug zur Gegenüberstellung verschiedener Planversionen

Die technologische Basis bildet ein cross-platform GUI-Framework mit nativer Desktop-Integration, leistungsfähigen PDF-Rendering-Fähigkeiten sowie einer umfangreichen Widget-Bibliothek für komplexe Benutzerinteraktionen.

### 6.7.3 PDF-Viewer mit Overlay-System

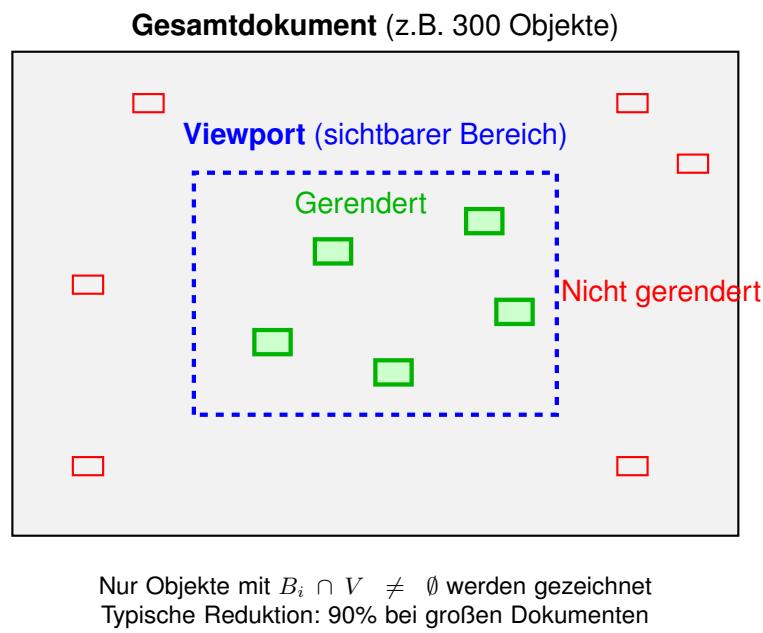
#### Rendering-Pipeline

Das PDF-Rendering erfolgt in zwei Schritten: Zunächst wird jede PDF-Seite bei hoher Auflösung (500 DPI) in ein Raster-Bild konvertiert. Die Anzeigebreite  $w_{display}$  berechnet sich aus der

Seitenbreite  $w_{\text{page}}$ , der Auflösung  $r$  und dem Zoomfaktor  $z$  gemäß:

$$w_{\text{display}} = w_{\text{page}} \times \frac{r}{72} \times z \quad (6.39)$$

wobei 72 die Standard-DPI-Referenz für PDF-Dokumente darstellt. Der Zoomfaktor  $z$  ist im Bereich [0.25, 4.0] steuerbar, was eine 16-fache Vergrößerungsspanne ermöglicht. Zur Performance-Optimierung wird ein Viewport-basiertes Culling implementiert (Abbildung 6.35), das die Anzahl zu rendernder Overlays auf die im aktuellen Sichtbereich befindlichen Objekte reduziert. Dies verbessert die Darstellungsgeschwindigkeit bei großformatigen Plänen erheblich.



**Abbildung 6.35:** Viewport-basiertes Culling zur Performance-Optimierung

### Overlay-Darstellung

Über dem gerenderten PDF wird eine transparente Grafikschicht gelegt, auf der die erkannten Bounding Boxes (OBBs) als Polygone visualisiert werden. Alle 12 Objektklassen werden einheitlich in Grün dargestellt, um eine klare visuelle Abgrenzung vom Dokumentinhalt zu gewährleisten. Jede Bounding Box zeigt zusätzlich folgende Informationen als Textlabel an:

- Objektbezeichnung (z. B. Signalname, GKS-Nummer)
- Konfidenzwert der Detektion
- Rotationswinkel der OBB
- Verknüpfte Koordinate (falls vorhanden)

### 6.7.4 Interaktive Ergebnistabelle

#### Hierarchische Datenstruktur

Die Ergebnisdarstellung erfolgt in einer Baumstruktur mit zweistufiger Hierarchie: Auf der ersten Ebene werden Elemente nach ihrer Klasse gruppiert, auf der zweiten Ebene befinden sich die individuellen Detektionen. Diese Strukturierung ermöglicht sowohl einen schnellen Überblick über die Klassenverteilung als auch detaillierte Einsicht in einzelne Erkennungen. Die Tabellenspalten umfassen:

1. **Klasse**: Objekttyp (Signal, Weiche, etc.)
2. **Text**: Extrahierter OCR-Text (Objekttext oder Koordinatennummer)
3. **Koordinate**: Zugeordneter Koordinatenwert
4. **Konfidenz**: YOLO-Detektionskonfidenz  $\in [0, 1]$
5. **Seite**: Seitennummer im PDF-Dokument
6. **Position**: Pixelkoordinaten  $(x, y)$

#### Sortierung und Filterung

Die Sortierung erfolgt spaltenweise über einen Vergleichsoperator, der numerische und lexicographische Ordnung unterscheidet. Für eine Spalte  $c$  und zwei Zeilen  $i, j$  mit Werten  $v_i^c, v_j^c$  gilt:

$$v_i^c \prec v_j^c \iff \begin{cases} v_i^c < v_j^c & \text{falls } v_i^c, v_j^c \in \mathbb{R} \\ \text{lexord}(v_i^c, v_j^c) & \text{sonst} \end{cases} \quad (6.40)$$

Die Filterung basiert auf regulären Ausdrücken. Gegeben sei ein Regex-Muster  $p$  und ein Zellenwert  $v$ , so wird die Zeile angezeigt, wenn:

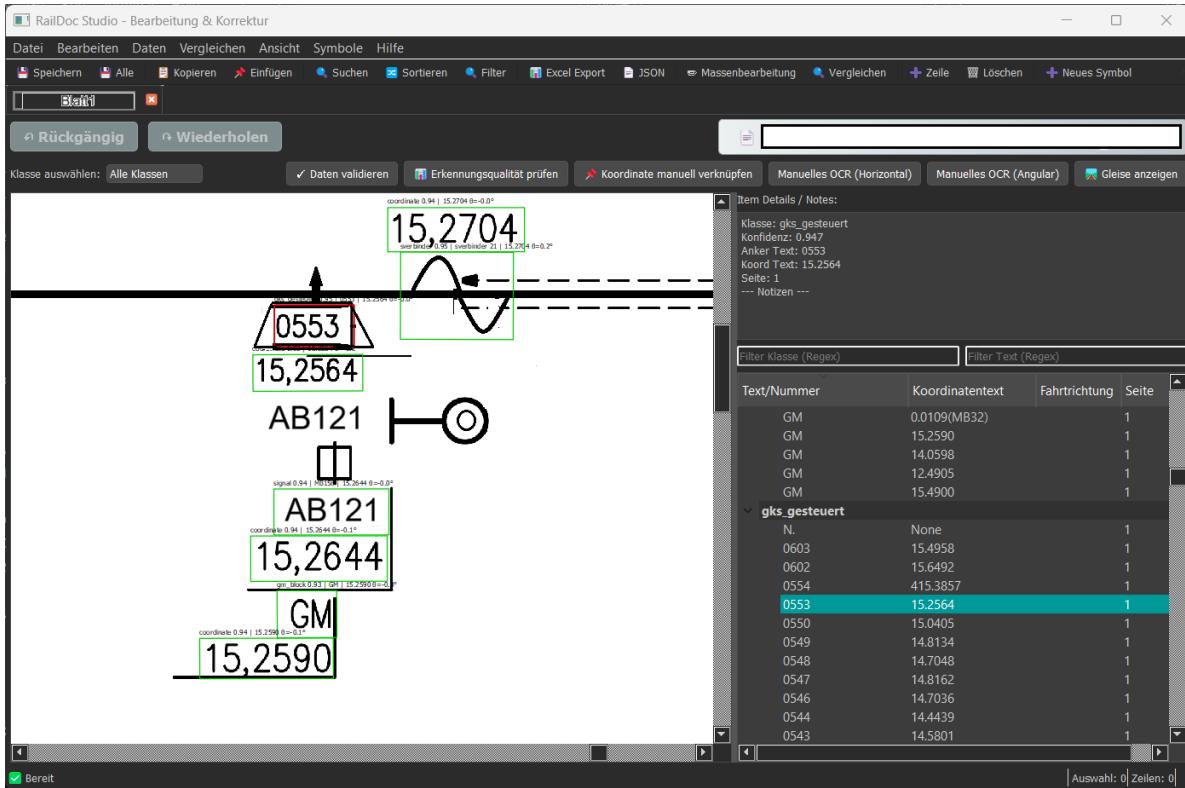
$$\exists c \in \{\text{Spalten}\} : p \text{ matched } v_c \quad (6.41)$$

#### Bidirektionale Verlinkung

Bei Auswahl einer Tabellenzeile wird die entsprechende Bounding Box in der PDF-Ansicht hervorgehoben. Die Navigation erfolgt durch Zentrierung des Viewports auf die Objektposition  $(x_c, y_c)$  mit einem Hervorhebungsradius  $r_{\text{highlight}} = 50$  Pixel. Die Hervorhebung wird als gelber Kreis mit zeitlich begrenzter Sichtbarkeit ( $t_{\text{highlight}} = 0.5$  Sekunden) dargestellt.

Die inverse Verlinkung (PDF  $\rightarrow$  Tabelle) erfolgt über einen eindeutigen Zeilen-Identifier, der sowohl im Tabellenelement als auch in der grafischen Repräsentation gespeichert wird. Bei einem Klick auf eine Bounding Box wird die zugehörige Tabellenzeile automatisch selektiert und

in den sichtbaren Bereich gescrollt. Abbildung 6.36 zeigt die vollständige Benutzeroberfläche mit allen beschriebenen Komponenten in integrierter Darstellung.



**Abbildung 6.36:** Hauptfenster mit PDF-Viewer (links), hierarchischer Ergebnistabelle (rechts) und bidirektionaler Navigation

Die Abbildung demonstriert die praktische Anwendung der bidirektionalen Verlinkung: Das in der Tabelle selektierte Objekt (GKS 0553 mit Koordinate 15,2564, hervorgehoben in cyan) ist im PDF-Viewer durch eine grüne Bounding Box markiert und zentriert. Die farbliche Codierung der Overlays folgt dem definierten Schema (Das selektierte Element wird rot hervorgehoben, während die übrigen Erkennungen grün dargestellt werden). Die Toolbar am oberen Rand bietet Funktionen zur Datenvierlidierung, Erkennungsqualitätsprüfung, manuellen Koordinatenverknüpfung und OCR-Re-Extraktion. Das rechte Panel zeigt Detailinformationen zum selektierten Objekt, einschließlich Klassentyp (gks\_gesteuert), YOLO-Konfidenz (0,947), extrahiertem Text und Seitenzuordnung. Diese integrierte Ansicht erfüllt die Anforderungen FA-012 (Visuelle Validierung) und FA-013 (GUI) und ermöglicht eine effiziente Qualitätsprüfung ohne Medienbruch zwischen Originaldokument und strukturierten Daten.

### 6.7.5 Validierungs-Dialog

Der Validierungs-Dialog implementiert eine mehrstufige Fehlerklassifikation basierend auf den Validierungsregeln aus Abschnitt 6.5. Die Fehlertypen umfassen:

- **Regex-Mismatch:** OCR-Text entspricht nicht dem erwarteten Format für die jeweilige Klasse

- **Niedrige Konfidenz:** YOLO-Konfidenz  $< \tau_{\text{conf}}$  (Standard:  $\tau_{\text{conf}} = 0.6$ )
- **Fehlende Verknüpfung:** Ankerelement ohne zugeordnete Koordinate
- **Duplikate:** Mehrfachvorkommen identischer Element-IDs

### Interaktionsmöglichkeiten

Für jeden identifizierten Fehler stehen folgende Korrekturoptionen zur Verfügung:

1. **Manuelle Korrektur:** Direkte Inline-Bearbeitung des Zellenwerts
2. **erneuter OCR Versuch:** Erneute OCR-Ausführung mit alternativen Parametern (horizontal/angular)
3. **Löschen:** Entfernung des fehlerhaften Eintrags
4. **Bestätigung:** Akzeptanz trotz Warnung (z.B. bei bekannten Sonderfällen)

### Transaktionsmodell

Änderungen werden zunächst in einer temporären Warteschlange gespeichert und erst bei Dialogbestätigung atomar in das Haupt-DataFrame übertragen. Dies gewährleistet Datenkonsistenz und ermöglicht vollständiges Rollback bei Abbruch. Die Änderungshistorie folgt dem Muster:

$$\Delta_{\text{validation}} = \{(r_i, f_i, v_{\text{old},i}, v_{\text{new},i}) \mid i = 1, \dots, n\} \quad (6.42)$$

wobei  $r_i$  die Zeilen-ID,  $f_i$  das Feld,  $v_{\text{old},i}$  den alten und  $v_{\text{new},i}$  den neuen Wert bezeichnet.

#### 6.7.6 Versionsvergleichs-Ansicht

##### Side-by-Side Darstellung

Der Versionsvergleich visualisiert zwei Gleispläne (Version A und Version B) nebeneinander in einem geteilten Anzeigebereich. Jede Version erhält ein eigenständiges PDF-Rendering mit korrespondierenden Overlays. Die Aufteilung des Anzeigebereichs ist dynamisch anpassbar, sodass der Benutzer die Größenverhältnisse nach Bedarf verändern kann.

##### Matching-Algorithmus

Die Zuordnung korrespondierender Elemente zwischen zwei Planversionen erfolgt mittels des *Hungarian Algorithm* (Kuhn-Munkres-Algorithmus), der eine global optimale Zuweisung gewährleistet. Im Gegensatz zu einem Greedy-Ansatz, der bei Duplikaten (mehrere Elemente derselben Klasse an ähnlichen Positionen) zu fehlerhaften Zuordnungen führen kann, findet der Hungarian-Algorithmus die Zuweisung mit maximalem Gesamtscore.

Das Matching erfolgt in zwei Phasen:

**Phase 1 – UUID-Matching:** Elemente mit identischer `detection_id` werden direkt zugeordnet, da diese eine eindeutige Identifikation garantieren.

**Phase 2 – Semantisches Fallback-Matching:** Für nicht-zugeordnete Elemente wird der Hungarian-Algorithmus klassenweise angewendet. Dabei wird eine Kostenmatrix  $C$  erstellt, wobei  $C_{ij} = -\text{score}(a_i, b_j)$  den negativen Ähnlichkeitsscore zwischen Element  $a_i$  der Altversion und Element  $b_j$  der Neuversion darstellt.

Die Scoring-Funktion berücksichtigt klassenspezifische Kriterien:

- **OCR-Klassen** (Signal, GKS): Primär Textähnlichkeit des Ankertextes, ergänzt durch Koordinaten- und Positionsboni. Kein Distanz-Cutoff, da eindeutige Bezeichner vorliegen.
- **Symbol-Klassen** (Sverbinder, GM-Block, etc.): Primär Koordinatenwert-Übereinstimmung mit hartem Cutoff bei  $\Delta_{\text{coord}} > 100 \text{ m}$ , da keine eindeutigen Bezeichner existieren.

Eine Zuordnung wird akzeptiert, wenn der Score den Schwellenwert  $\theta = 0,7$  überschreitet. Nicht zugeordnete Elemente der Altversion gelten als *gelöscht*, nicht zugeordnete Elemente der Neuversion als *hinzugefügt*.

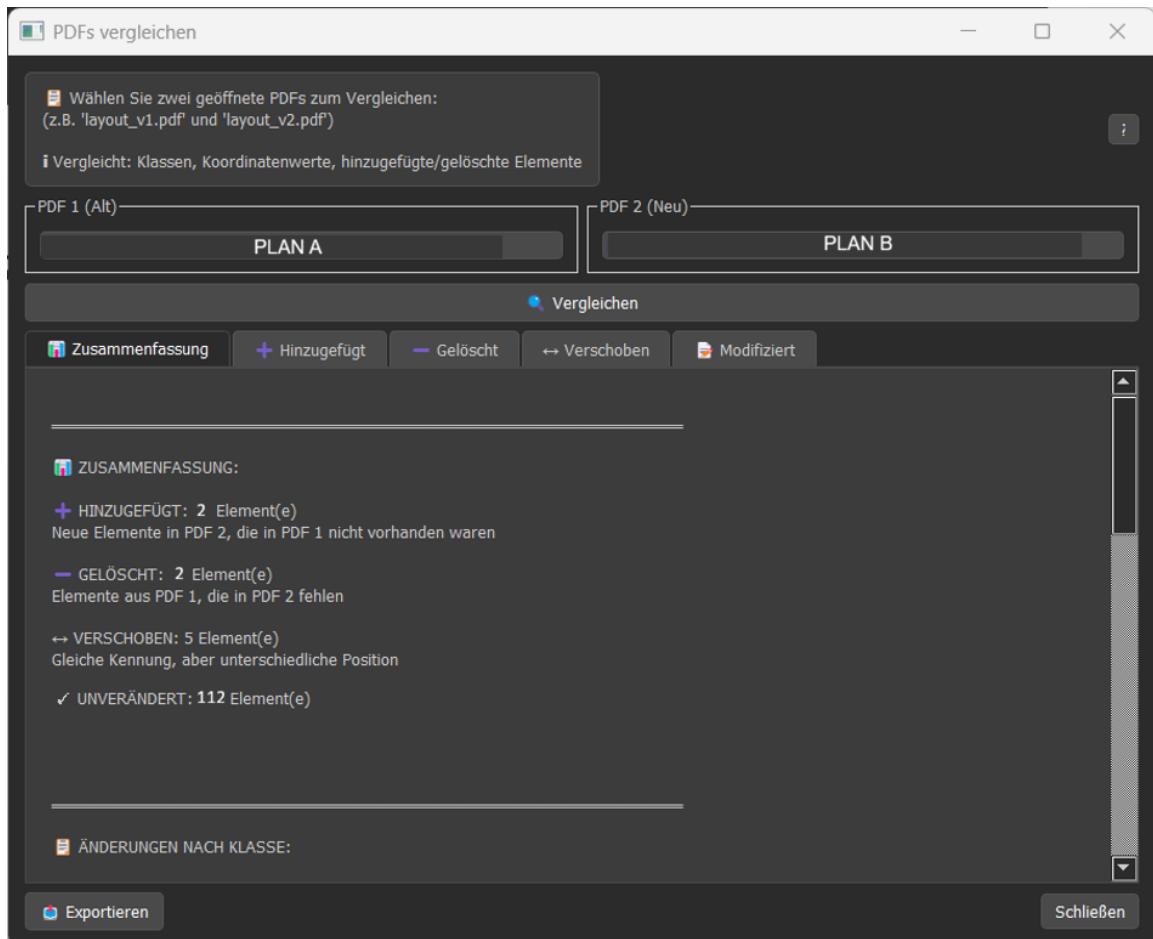
Ein Element gilt als *verschoben*, wenn sich dessen Kilometrierungswert (`coord_value`) zwischen den Versionen geändert hat. Die Schwere der Verschiebung wird nach Distanz klassifiziert: geringfügig ( $< 5 \text{ m}$ ), moderat ( $5\text{--}20 \text{ m}$ ) oder signifikant ( $> 20 \text{ m}$ ).

## Synchronisierte Navigation

Die Scrollbars beider PDF-Ansichten sind gekoppelt, sodass ein Scrollen in einem Viewer automatisch den anderen Viewer mitbewegt. Die Synchronisation erfolgt über Signal-Slot-Verbindungen:

$$s_B = s_A \cdot \frac{h_B}{h_A} \quad (6.43)$$

wobei  $s_A$  die Scrollposition in Viewer A,  $h_A$  dessen Gesamthöhe und entsprechend für Viewer B. Dies gewährleistet proportionale Navigation bei unterschiedlichen Seitengrößen.



**Abbildung 6.37:** Änderungsvergleich-Dialog

Abbildung 6.37 zeigt den Dialog zum Vergleich zweier Planversionen. Nach Auswahl der zu vergleichenden PDFs (PDF 1 als Referenz, PDF 2 als aktualisierte Version) analysiert das System automatisch alle Unterschiede. Die Zusammenfassung kategorisiert die Ergebnisse in drei Änderungstypen: *Hinzugefügt* (neue Elemente in PDF 2), *Gelöscht* (fehlende Elemente in PDF 2) und *Verschoben* (gleiche Kennung, unterschiedliche Kilometrierung). Zusätzlich werden *Unveränderte* Elemente angezeigt. Im dargestellten Beispiel wurden zwischen zwei Versionen des Plans insgesamt 41 Änderungen erkannt: 21 hinzugefügte, 15 gelöschte und 5 verschobene Elemente. Die einzelnen Tabs ermöglichen die detaillierte Inspektion jeder Änderungskategorie.

### 6.7.7 Export-Funktionalität

#### Excel-Export

Der Excel-Export erzeugt strukturierte Arbeitsblätter mit automatischer Spaltenformatierung. Jede Objektklasse erhält ein eigenes Tabellenblatt mit klassenspezifischen Spalten.

Zusätzlich wird ein *Metadata-Sheet* erzeugt, das Verarbeitungsmetadaten enthält:

- Verarbeitungsdatum und -zeitpunkt

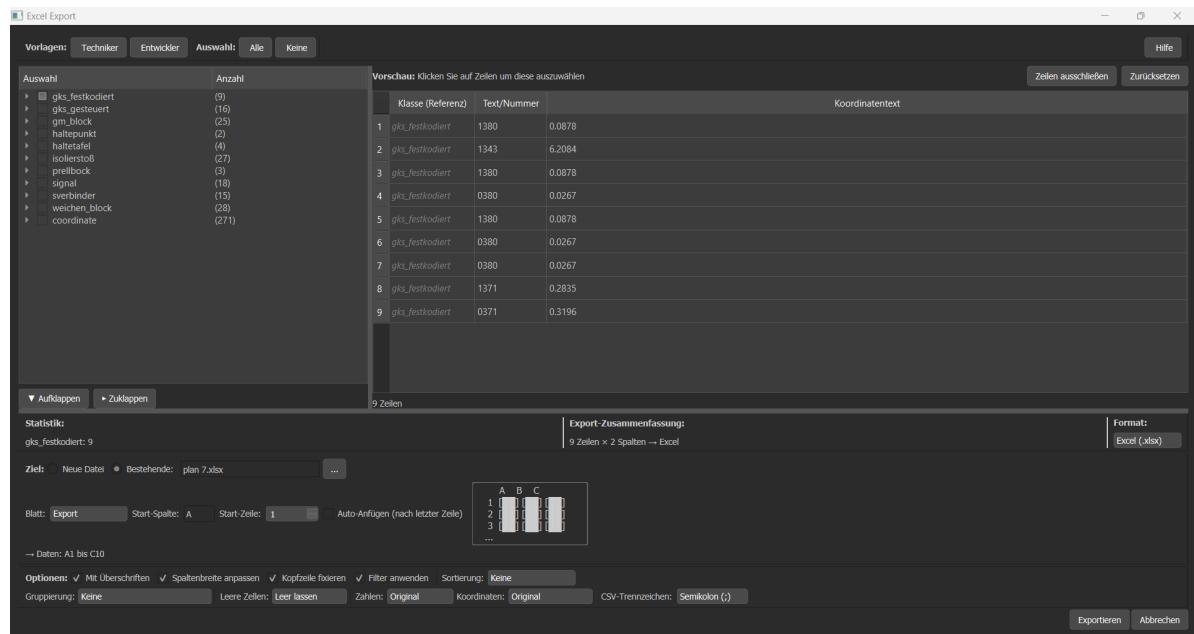
- YOLO-Modellversion und -pfad
- Konfidenzstatistiken:  $\mu_{\text{conf}}$ ,  $\sigma_{\text{conf}}$ ,  $\min(\text{conf})$ ,  $\max(\text{conf})$
- Anzahl Detektionen pro Klasse

## Erweiterter Export-Dialog

Für die praktische Anwendung wurde ein umfassender Export-Dialog implementiert (Abbildung 6.38), der über die Grundfunktionalität hinausgeht und die Anforderungen FA-009 (Excel-Integration) sowie FA-010 (Strukturerhalt) adressiert. Die wichtigsten Funktionen umfassen:

- **Vorlagen:** Zwei vordefinierte Spaltenauswahlen für unterschiedliche Anwendergruppen:
  - *Techniker*: Reduzierte Ansicht mit Klassenname, Koordinate und Fahrtrichtung – optimiert für Bahntechniker, die nur die betriebsrelevanten Daten benötigen
  - *Entwickler*: Vollständige Ansicht mit allen technischen Details (Bounding-Box-Koordinaten, Rotationswinkel, Konfidenzwerte) – für KI-Entwickler zur Modellvalidierung
- **Klassenbasierte Auswahl:** Hierarchische Baumansicht aller Objektklassen mit Anzahl der Instanzen; selektiver Export einzelner Klassen
- **Live-Vorschau:** Tabellarische Vorschau der zu exportierenden Daten mit Möglichkeit zum Ausschließen einzelner Zeilen
- **Zieloptionen:** Export in neue Datei oder Anhängen an bestehende Excel-Datei mit wählbarer Startposition (Blatt, Spalte, Zeile)
- **Formatierung:** Optionen für Überschriften, Spaltenbreiten, fixierte Kopfzeile, Zahlenformat und Leerzellenbehandlung
- **Gruppierung:** Export aller Klassen in ein Blatt, als Sektionen, oder auf separate Arbeitsblätter (vgl. Abbildung 6.39)

Die ASCII-Vorschau der Zielposition (rechts unten in Abbildung 6.38) visualisiert, wo die Daten in einer bestehenden Datei eingefügt werden, und verhindert versehentliches Überschreiben vorhandener Inhalte.



**Abbildung 6.38:** Export-Dialog mit klassenbasierter Auswahl, Live-Vorschau und Zielpositionierung für bestehende Dateien

Abbildung 6.39 zeigt ein Beispiel der exportierten Daten mit separaten Arbeitsblättern pro Objektklasse. Die resultierenden Excel-Dateien dienten als Grundlage für den Ground-Truth-Vergleich der E2E-Evaluation (vgl. Tabelle 7.6).

Die Option „Bestehende Datei“ (Abbildung 6.38) ermöglicht das Einfügen von Daten in existierende Excel-Vorlagen ab einer wählbaren Startposition, ohne bestehende Formatierungen, Formeln oder Makros zu überschreiben. Diese Funktionalität erfüllt die Anforderung FA-010 (Strukturerhalt).

	A	B	C	D	E	F	G	H	I	J	K	L
5	1231	0.1682										
6	1530	13.6474										
7	1510	11.9476										
8	7561	14.9915										
9	5722	14.8056										
10	1222	14.7157										
11	2312	14.2917										
12	4565	0.1708										
13	4524	0.2043										
14	1234	0.1483										
15	1232	0.1844										
16	2131	14.5348										
17	AB456	0.0266	A									
18	AB254	14.6966	B									

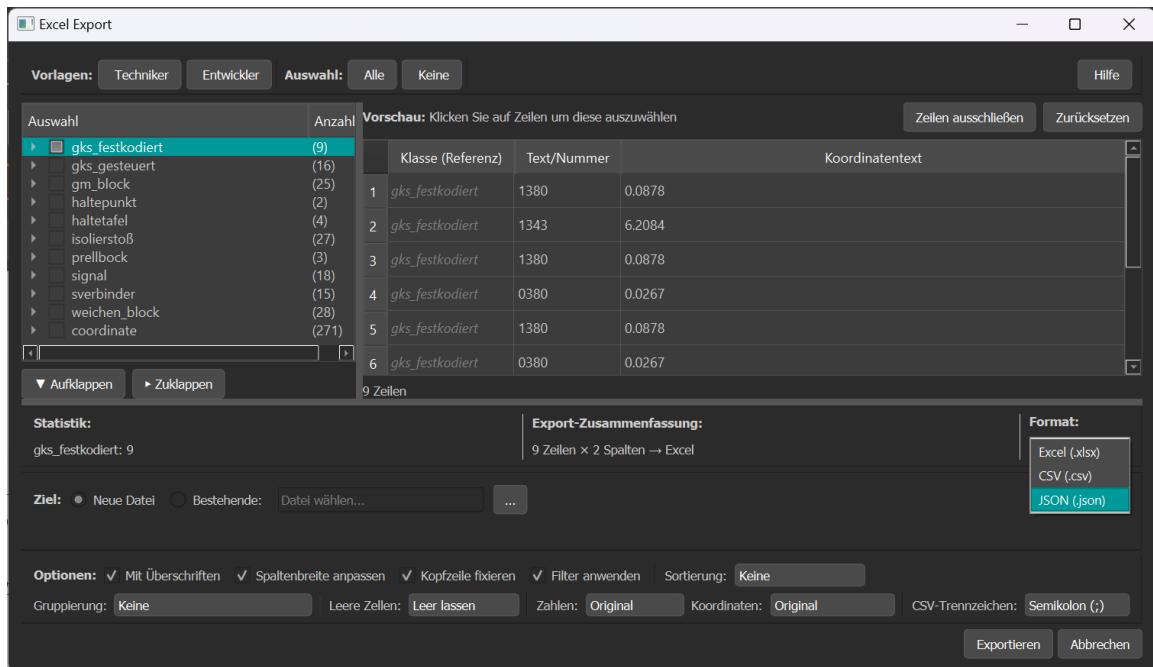
**Abbildung 6.39:** Exportierte Excel-Datei mit separaten Arbeitsblättern pro Objektklasse  
(sichtbar am unteren Rand: gks\_festkodiert, gks\_gesteuert, gm\_block, etc.)

## JSON-Export

Der JSON-Export folgt einem hierarchischen Schema mit vollständigen Geometriedaten:

```
{
  "elements": [
    {
      "id": <einheitige ID>,
      "class": <Objektklasse>,
      "text": <OCR-Text>,
      "Konfidenz": <float in [0,1]>,
      "obb_points": [[x1,y1], [x2,y2], [x3,y3], [x4,y4]],
      "page": <Seitennummer>,
      "linked_elements": [<IDs verknüpfter Objekte>],
      "ocr_engine": <verwendete OCR-Engine>,
      "linking_method": <Verknüpfungsmethode>
    }
  ]
}
```

Diese Struktur ermöglicht die vollständige Rekonstruktion der Extraktionsergebnisse und eignet sich für Weiterverarbeitung in anderen Systemen.



**Abbildung 6.40:** Export-Dialog mit Formatauswahl. Das Dropdown-Menü (rechts) bietet drei Ausgabeformate: Excel (.xlsx), CSV (.csv) und JSON (.json).

Gemäß Anforderung **NFA-010** unterstützt das System drei Ausgabeformate:

- **Excel (.xlsx):** Primärformat für Engineering-Workflows mit Unterstützung für Formatierung, mehrere Arbeitsblätter und Formelerhalt
- **CSV (.csv):** Textbasiertes Format für einfachen Datenaustausch und Import in Drittsysteme
- **JSON (.json):** Maschinenlesbares Format für API-Anbindung; die Struktur ist konfigurierbar (Records als Liste von Objekten oder spaltenbasiertes Array)

Die Formatauswahl erfolgt über ein Dropdown-Menü im Export-Dialog (Abbildung 6.40).

### Export des Differenzberichts

Die erkannten Änderungen können über die Schaltfläche *Exportieren* in eine Excel-Datei ausgegeben werden (Abbildung 6.41). Die exportierte Arbeitsmappe enthält separate Arbeitsblätter für jede Änderungskategorie (Zusammenfassung, Hinzugefügt, Gelöscht, Verschoben), wodurch eine strukturierte Dokumentation der Planänderungen für Revisionszwecke ermöglicht wird. Bei verschobenen Elementen werden sowohl der alte als auch der neue km-Wert sowie die Differenz in Metern angegeben.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Objektklasse	Kennung	Änderungstyp	Alt-Wert	Neu-Wert	Koordinate	Seite							
2	gks_gesteuert	0305	Verschoben	0.1770	0.1729 (- 4.1m)	0.1729	1							
3	gks_gesteuert	0303	Verschoben	0.1470	0.1495 (+2.5m)	0.1495	1							
4	gks_gesteuert	0303	Verschoben	0.1470	0.1495 (+2.5m)	0.1495	1							
5	gks_gesteuert	0305	Verschoben	0.1770	0.1729 (- 4.1m)	0.1729	1							
6	gks_gesteuert	0303	Verschoben	0.1470	0.1495 (+2.5m)	0.1495	1							

Abbildung 6.41: Änderungsexport

Wie bereits in Abbildung 6.29 erläutert, können GKS-Kennungen mehrfach auftreten, wenn der Gleisplan mehrere Streckenabschnitte enthält. Die exportierte Tabelle spiegelt diese Struktur wider: GKS 0303 erscheint dreimal und GKS 0305 zweimal, entsprechend ihrer Instanzen in den verschiedenen Planabschnitten.

## 6.8 Zusammenfassung

Dieses Kapitel stellte die technische Realisierung der automatisierten Datenextraktionspipeline für Gleispläne vor. Die Implementierung umfasst drei zentrale Verarbeitungsstufen sowie unterstützende Komponenten für Qualitätssicherung, Persistenz und Benutzerinteraktion.

### 6.8.1 Kernkomponenten der Pipeline

Die Extraktionspipeline besteht aus folgenden Hauptmodulen:

**Objekterkennung (Abschnitt 6.2):** Das YOLOv8-OBB Modell wurde auf einem Datensatz mit 13 Symbolklassen trainiert (vgl. Anforderung FA-003). Die Klasse *coordinate* liefert Positionsinformationen, die den anderen 12 Klassen zugeordnet werden. Alle 13 Klassen werden in Kapitel 7 quantitativ evaluiert.

**OCR-Pipeline (Abschnitt 6.3):** Die orientierungsadaptive OCR-Pipeline bildet das Herzstück der Textextraktion. Durch die Kaskadierung dreier OCR-Engines (PaddleOCR, Tesseract, EasyOCR) wird eine hohe Erkennungsrobustheit erreicht. Das Dual-Winkel-Routing-System behandelt sowohl achsenparallele als auch beliebig rotierte Texte mittels unterschiedlicher Transformationsstrategien. Klassenspezifische Vorverarbeitungspipelines passen die Bildaufbereitung an die jeweiligen Charakteristika der Symbole an (z.B. adaptive Padding für Signale, morphologische

Filterung für GKS). Die Validierung erfolgt durch regex-basierte Pattern-Matching-Verfahren mit konfidenzgewichteten Scoring-Mechanismen.

**Symbol-Text-Verknüpfung (Abschnitt 6.4):** Die intelligente Linking-Komponente ordnet extrahierte Texte ihren zugehörigen Symbolen zu. Die rotationsinvariante Koordinatentransformation ermöglicht eine geometrisch konsistente Richtungsbeziehung unabhängig von der Symbolorientierung. Der proximity-basierte Algorithmus nutzt klassenspezifische Suchradien und berücksichtigt erwartete räumliche Relationen. Ein adaptiver Lernmechanismus erfasst wiederkehrende Layoutmuster und optimiert die Verknüpfung durch probabilistische Fenstersuche. Speziallogiken behandeln komplexe Fälle wie die geometrische Ableitung der Fahrtrichtung aus der GKS-Signal-Relation oder die Gruppierung von Haltepunkt-Tripeln.

### 6.8.2 Unterstützende Systeme

**Validierung und Qualitätssicherung (Abschnitt 6.5):** Ein mehrstufiges Validierungsframework prüft extrahierte Daten syntaktisch (regex-basiert), semantisch (Plausibilitätsbereiche) und referentiell (Vollständigkeit von Relationen). Automatische Korrekturen behandeln häufige OCR-Fehler, während fehlerhafte Einträge zur manuellen Überprüfung gekennzeichnet werden. Die Integration mit der UI ermöglicht einen interaktiven Validierungsworkflow.

**Versionsvergleich und Rückverfolgbarkeit (Abschnitt 6.6):** Die Versionsvergleichskomponente verwendet den Hungarian-Algorithmus zur optimalen Zuordnung von Elementen zwischen Planrevisionen. Änderungen werden in drei Kategorien klassifiziert: hinzugefügt, entfernt und verschoben (basierend auf Kilometrierungsänderungen). Das Rückverfolgbarkeitssystem persistiert für jedes extrahierte Element die Quellkoordinaten und PDF-Metadaten, wodurch eine bidirektionale Navigation zwischen Datenansicht und Plandarstellung ermöglicht wird.

**Benutzeroberfläche (Abschnitt 6.7):** Die PyQt5-basierte Benutzeroberfläche integriert die Pipeline-Komponenten in einem interaktiven Workflow. Der PDF-Viewer visualisiert Detektionen mittels farbcodierter Overlays und unterstützt Click-to-Highlight-Navigation. Die Ergebnistabelle bietet Sortier- und Filterfunktionen sowie Inline-Editing für Korrekturen. Der Validierungsdialog strukturiert fehlerhafte Einträge nach Fehlertyp und ermöglicht Batch-Operationen. Die Vergleichsansicht stellt Änderungen zwischen Planversionen side-by-side dar.

### 6.8.3 Implementierungsumfang

Die Gesamtimplementierung umfasst folgende Modulgrößen:

Modul	Umfang (LOC)
Objekterkennung	~2.000
OCR-Pipeline	~3.500
Linking-Algorithmen	~1.800
Validierung & Export	~900
Versionsvergleich	~650
Rückverfolgbarkeit	~400
UI-Komponenten	~2.100
<b>Gesamt</b>	<b>~11.350</b>

**Tabelle 6.14:** Codeumfang der Implementierungsmodule (gerundet)

Die Architektur folgt einer modularen Struktur, die eine klare Trennung der Verarbeitungsstufen gewährleistet. Zentrale Design-Entscheidungen (vgl. Kapitel 5) wurden konsequent umgesetzt:

- **Kaskadierte Fehlerbehandlung:** Jede Verarbeitungsstufe implementiert Fallback-Mechanismen zur Behandlung von Randfällen.
- **Klassenspezifische Parametrisierung:** Schwellenwerte, Suchradien und Preprocessing-Parameter sind pro Symbolklasse konfigurierbar.
- **Stateless Processing:** Die Kernpipeline ( $YOLO \rightarrow OCR \rightarrow Linking$ ) arbeitet zustandslos, wodurch parallele Verarbeitung erleichtert wird.
- **Metadaten-Tracking:** Alle Extraktionsschritte dokumentieren ihre Entscheidungen (verwendete Engine, Konfidenz-Werte, Transformationsparameter) für Transparenz und Debugging.

#### 6.8.4 Deployment und Ausführung

Das System wurde als Standalone-Desktop-Anwendung konzipiert, die keine Cloud-Infrastruktur zur Laufzeit benötigt. Dies gewährleistet Datenschutz-Compliance bei der Verarbeitung sensibler Bahninfrastrukturdaten. Die Modellinferenz erfolgt auf lokaler GPU-Hardware (oder CPU-Fallback), während die Datenpersistenz optional über eine lokale PostgreSQL-Instanz realisiert wird.

Die gesamte Verarbeitungszeit für einen typischen Gleisplan variiert je nach Komplexität:

$$T_{total} = T_{YOLO} + \sum_{i=1}^N T_{OCR_i} + T_{linking} + T_{validation} \quad (6.44)$$

wobei  $N$  die Anzahl der detektierten Objekte bezeichnet und  $T_{OCR_i}$  die klassenspezifische OCR-Dauer für Objekt  $i$  repräsentiert. Eine detaillierte Performanz-Analyse mit konkreten Zeitmessungen erfolgt in Kapitel 7.

### 6.8.5 Schnittstellen und Erweiterbarkeit

Die Implementierung definiert klare Schnittstellen zwischen den Modulen:

- **Detection Interface:** YOLO liefert strukturierte Objekte mit Klasse, OBB-Koordinaten, Rotationswinkel und Konfidenz.
- **OCR Interface:** Die OCR-Komponente akzeptiert Bildausschnitte und Klassenkontakte, liefert Text und Engine-Metadaten.
- **Linking Interface:** Der Linking-Algorithmus konsumiert Detektions- und OCR-Ergebnisse, produziert Assoziationen mit Konfidenz-Scores.
- **Validation Interface:** Das Validierungssystem prüft verknüpfte Objekte gegen konfigurierbare Regelsets.

Diese Modularität ermöglicht die zukünftige Integration alternativer Modelle (z.B. neuerer YOLO-Versionen) oder zusätzlicher OCR-Engines ohne Anpassung der nachgelagerten Verarbeitungslogik.

### 6.8.6 Abschließende Bemerkungen

Die vorgestellte Implementierung realisiert eine vollständige Pipeline von der PDF-Eingabe bis zum strukturierten Datenexport. Die Kombination aus lernbasierten Komponenten (YOLO, neuronale OCR-Engines), algorithmischen Verfahren (Koordinatentransformation, Proximity-Search) und regelbasierten Validierungen schafft ein robustes System für die Extraktion von Bahnanlagendaten aus technischen Zeichnungen.

Die technische Evaluation der implementierten Komponenten, einschließlich Genauigkeitsmetriken, Fehleranalysen und Performanzmessungen, wird im folgenden Kapitel 7 präsentiert.

# 7. Evaluation

Dieses Kapitel präsentiert die systematische Evaluation des entwickelten Prototyps. Die Bewertung erfolgt anhand definierter Metriken und validiert die in Kapitel 4 spezifizierten funktionalen und nicht-funktionalen Anforderungen. Die Evaluation gliedert sich in die Beschreibung der Testmethodik, die detaillierte Analyse der Einzelkomponenten sowie die Bewertung der Gesamt-systemleistung auf einem unabhängigen Testsatz.

## 7.1 Testmethodik

Die Evaluation des Prototyps erfordert eine sorgfältige Definition der Testbedingungen, um reproduzierbare und aussagekräftige Ergebnisse zu gewährleisten. Dieser Abschnitt beschreibt die verwendeten Testdatensätze, die angewandten Evaluationsmetriken sowie die Testumgebung.

### 7.1.1 Testdatensätze

Zur systematischen Bewertung wurden zwei hierarchisch strukturierte Datensätze verwendet, die verschiedene Evaluationszwecke erfüllen.

#### Validierungssatz (YOLO Technical Validation)

Der Validierungssatz dient der technischen Bewertung der Objekterkennungskomponente und wurde während des Trainingsprozesses zur Modellselektion verwendet. Die detaillierte Beschreibung der Datensatzerstellung, Annotation und Augmentation findet sich in Abschnitt 6.2.1.

Der Datensatz umfasst 208 Validierungs-Tiles mit 3.076 annotierten Symbolinstanzen über alle 13 Symbolklassen (vgl. Tabelle 6.3).

**Verwendungszweck:** Der Validierungssatz dient der technischen Bewertung der YOLO-Dektionsleistung (mAP, Precision, Recall) und bestätigt die erfolgreiche Modellkonvergenz. Der Validierungssatz stellt keine vollständig unabhängige Testmenge dar; die unabhängige End-to-End-Evaluation erfolgt auf dem separaten Testsatz (Abschnitt 7.2.2).

#### Testsatz (End-to-End System Evaluation)

Für die Evaluation der vollständigen Extraktionspipeline (Detektion → OCR → Linking → Validierung) wurde ein Testsatz aus realen Siemens Mobility Gleisplänen ausgewählt.

**Datensatztrennung:** Die neun Testpläne wurden zufällig aus einem separaten Datenpool ausgewählt, der zu keinem Zeitpunkt für Training oder Validierung des YOLO-Modells verwendet wurde. Diese strikte Trennung gewährleistet eine vollständig unabhängige Evaluation der Systemleistung.

**Methodische Einschränkung:** Aufgrund des erheblichen manuellen Aufwands zur Erstellung vollständiger Ground-Truth-Daten für A0-Gleispläne (durchschnittlich 2-3 Stunden pro Plan) sowie der begrenzten Verfügbarkeit weiterer ungewöhnlicher Pläne wurden neun Pläne unterschiedlicher Komplexität für die End-to-End-Evaluation ausgewählt. Diese Pläne repräsentieren verschiedene Komplexitätsstufen und ermöglichen eine realistische Bewertung der Systemleistung. Für eine vollständig unabhängige Evaluation wären zusätzliche, komplett ungewöhnliche Pläne wünschenswert gewesen, was im Zeitrahmen dieser Masterarbeit jedoch nicht realisierbar war.

**Evaluationsumfang:** Die End-to-End-Evaluation umfasst alle 12 Symbolklassen, die mit Koordinaten verknüpft werden. Die Klasse *coordinate* liefert Positionsinformationen und wird als verknüpftes Attribut der anderen Klassen bewertet – d.h. eine Koordinate gilt als korrekt, wenn sie dem richtigen Symbol zugeordnet wurde.

Attribut	Wert
Anzahl Gleispläne	9
Seitengröße	A0 (841 × 1189 mm)
Seiten pro Plan	1
Herkunft	Siemens Mobility Projekte
Komplexitätsstufen	Einfach (2), Mittel (3), Komplex (4)
<i>Durchschnittliche Symbolanzahl pro Plan (12 Ankerklassen)</i>	
Signale	28
GM-Blöcke	27
GKS gesteuert	22
S-Verbinder	23
Isolierstöße	20
Weichen-Blöcke	19
GKS festkodiert	9
Haltepunkte	8
Weichenenden	3
Weichengruppenenden	1
Prellböcke	1
Haltetafeln	1
<b>Summe (12 Ankerklassen)</b>	<b>164</b>
Gesamtanzahl evaluierter Objekte	1473
Tiles pro Plan (Durchschnitt)	≈ 40
Auflösungsbereich	500 DPI

**Tabelle 7.1:** Charakteristika des Testdatensatzes (A0-Gleispläne)

**Ground-Truth-Erstellung:** Für jeden Testplan wurde eine manuelle Referenzdatei erstellt, die alle relevanten Extraktionsziele enthält:

- **Signale:** Bezeichnung (z.B. “A102”), zugehörige Kilometrierung, Fahrtrichtung (A/B)
- **GKS:** Nummer (z.B. “1234”), zugehörige Kilometrierung
- **Koordinatenangaben:** Kilometerwert (z.B. “18.1606”), optionale Gleisangabe

- **Verknüpfungen:** Erwartete Assoziationen zwischen Symbolen und Texten

Die Ground-Truth-Daten wurden in strukturierten Excel-Dateien gespeichert, um einen direkten Vergleich mit den Systemausgaben zu ermöglichen.

**Verwendungszweck:** Der Testsatz dient der vollständigen End-to-End-Evaluation aller Pipeline-Komponenten und misst die tatsächliche Systemleistung auf realen Daten aus dem Siemens Mobility Umfeld.

*Hinweis: Aus Vertraulichkeitsgründen (Sperrvermerk) werden keine spezifischen Projektbezeichnungen oder Visualisierungen der Originalpläne präsentiert. Die Ergebnisse werden in aggregierter Form berichtet.*

### Komplexitätskategorisierung

Um die Robustheit des Prototyps unter verschiedenen Bedingungen zu evaluieren, wurden die Testpläne in drei Komplexitätskategorien eingeteilt. Da alle Pläne im A0-Format vorliegen und jeweils eine Seite umfassen, erfolgt die Kategorisierung primär nach Symboldichte und Layoutkomplexität.

Kategorie	Charakteristik	Anzahl
Einfach	Niedrige Symboldichte (< 110 Symbole), klare räumliche Trennung, typisch für Streckenabschnitte	2
Mittel	Moderate Symboldichte (100–130 Symbole), gelegentliche Überlappungen, typisch für kleinere Bahnhöfe	3
Komplex	Hohe Symboldichte (> 180 Symbole), viele überlappende Elemente, dichte Weichenbereiche, typisch für große Bahnhofsköpfe	4

**Tabelle 7.2:** Komplexitätskategorien der A0-Testpläne

Diese Kategorisierung ermöglicht eine differenzierte Analyse der Systemleistung in Abhängigkeit von der Plankomplexität und identifiziert kritische Schwellenwerte für Symboldichte und räumliche Überlappung.

#### 7.1.2 Evaluationsmetriken

Die Bewertung des Prototyps erfolgt auf mehreren Ebenen mit jeweils spezifischen Metriken. Die verwendeten Metriken basieren auf den in Kapitel 3 eingeführten Standardverfahren für Objekterkennung (Abschnitt 3.1.7) und OCR-Systeme (Abschnitt 3.3.4). Dieser Abschnitt fasst die angewandten Metriken kurz zusammen und definiert die spezifische End-to-End Systemmetrik.

## Metriken für die Objekterkennung

Für die Bewertung der YOLO-basierten Objekterkennung werden die in Abschnitt 7.1.2 definierten Standardmetriken verwendet:

- **Precision:** Anteil korrekter Detektionen an allen Vorhersagen
- **Recall:** Anteil gefundener Objekte an allen vorhandenen Objekten
- **F1-Score:** Harmonisches Mittel aus Precision und Recall
- **mAP@0.5:** Mean Average Precision bei IoU-Schwelle von 50%
- **mAP@0.5:0.95:** mAP gemittelt über IoU-Schwellen von 50% bis 95%

Eine Detektion gilt als *True Positive*, wenn die Intersection over Union (IoU) mit der Ground-Truth-Box  $\geq 0.5$  beträgt und die Klassenvorhersage korrekt ist.

## Metriken für die Texterkennung

Die OCR-Leistung wird nicht isoliert durch zeichenbasierte Metriken wie die Character Error Rate (CER) bewertet, sondern nach dem Prinzip der *Feldgenauigkeit* (vgl. Abschnitt 3.3.4): Ein OCR-Ergebnis gilt als korrekt, wenn der extrahierte Text exakt mit dem Ground Truth übereinstimmt. Diese Bewertung ist in die End-to-End-Systemmetrik integriert, wodurch folgende Vorteile entstehen:

- OCR-Fehler, die durch nachgelagerte Validierung (Regex-Muster) automatisch korrigiert werden, beeinflussen das Endergebnis nicht negativ
- Die Metrik entspricht dem tatsächlichen Informationsbedarf: „Wurde der korrekte Wert extrahiert? „statt“ Wie viele Zeichen waren falsch?“
- Fehlerquellen können der jeweiligen Pipeline-Stufe zugeordnet werden (YOLO vs. OCR vs. Linking)

Zusätzlich wird die **Regex-Validierungsrate** erfasst, die den Anteil der OCR-Ergebnisse quantifiziert, die klassenspezifische Formatmuster erfüllen.

## End-to-End Systemmetrik

Die Gesamtsystemleistung wird durch die **End-to-End Accuracy** gemessen, die dem in Anforderung **NFA-003** definierten Zielwert entspricht:

$$\text{E2E Accuracy} = \frac{\text{Vollständig korrekt extrahierte Objekte}}{\text{Gesamtzahl Objekte}} \times 100\% \quad (7.1)$$

Ein Objekt gilt als „vollständig korrekt extrahiert“, wenn alle folgenden Bedingungen erfüllt sind:

1. Das Symbol wurde korrekt detektiert ( $\text{IoU} \geq 0.5$  mit Ground Truth)

2. Die Klassifikation ist korrekt
3. Der OCR-Text stimmt exakt mit dem Ground Truth überein (falls anwendbar)
4. Alle erforderlichen Verknüpfungen (z.B. zu Koordinaten) sind korrekt (falls anwendbar)

### 7.1.3 Testumgebung

Die Evaluation wurde auf einer standardisierten Hardware- und Softwarekonfiguration durchgeführt, um reproduzierbare Ergebnisse zu gewährleisten. Tabelle 7.3 fasst die technischen Spezifikationen zusammen.

Komponente	Spezifikation
<i>Hardware (Inferenz)</i>	
CPU	AMD Ryzen 5 PRO 5650U (6 Kerne, 2.3 GHz)
RAM	32 GB DDR4
GPU	Keine (CPU-only Inferenz)
Speicher	512 GB SSD
<i>Hardware (Training)</i>	
GPU	NVIDIA T4 (AWS g4dn.xlarge)
VRAM	16 GB
<i>Software</i>	
Betriebssystem	Windows 10 / Ubuntu 22.04
Python	3.9.16
PyTorch	2.0.1
Ultralytics	8.0.196
PaddleOCR	2.7.0
Tesseract	5.3.0
PostgreSQL	14.9

**Tabelle 7.3:** Hardware- und Softwarekonfiguration der Testumgebung

Die Wahl einer CPU-basierten Inferenzumgebung reflektiert die Anforderung **NFA-001**, die eine On-Premise-Verarbeitung auf Standard-Workstations ohne dedizierte GPU vorsieht. Alle Zeitmessungen wurden als Mittelwert über drei Durchläufe berechnet, um Varianz durch Systemlast zu minimieren.

## 7.2 Ergebnisanalyse

Dieser Abschnitt präsentiert die quantitativen Evaluationsergebnisse der einzelnen Pipeline-Komponenten sowie des Gesamtsystems. Die Ergebnisse werden im Kontext der in Kapitel 4 definierten Anforderungen interpretiert.

### 7.2.1 Objekterkennungsleistung

Die Objekterkennung bildet die fundamentale Stufe der Extraktionspipeline. Die Qualität der YOLO-Detektionen determiniert maßgeblich die erreichbare Gesamtgenauigkeit des Prototyps.

### Gesamtleistung auf dem Validierungssatz

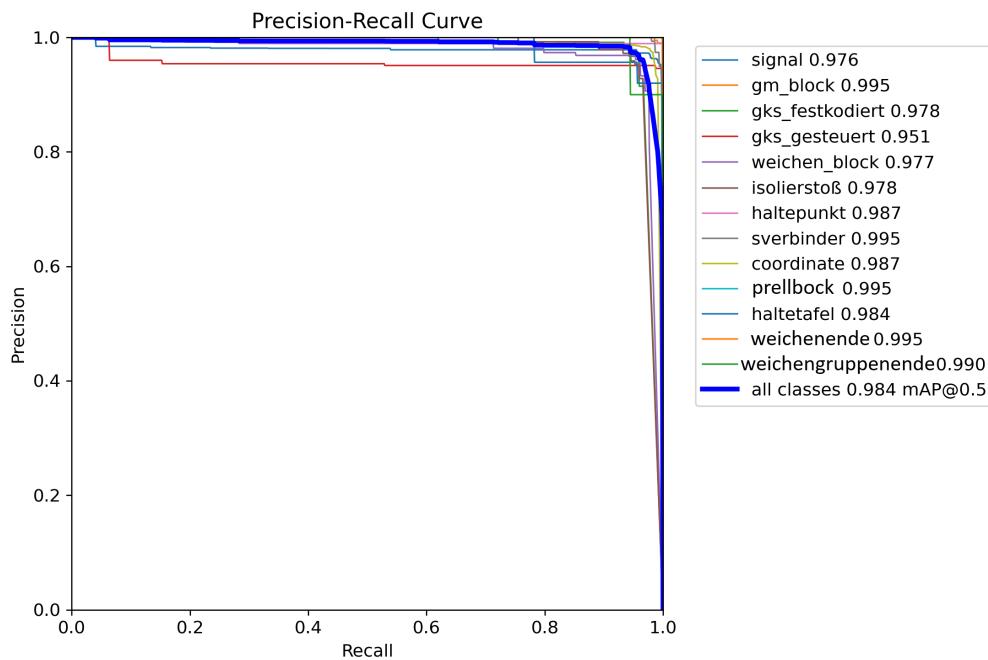
Das trainierte YOLOv8l-OBB Modell wurde auf dem Validierungsdatensatz (208 Bilder, 3.076 Instanzen) evaluiert. Tabelle 7.4 zeigt die aggregierten Metriken über alle Symbolklassen.

Metrik	Wert
Precision (Durchschnitt)	97.5%
Recall (Durchschnitt)	95.7%
F1-Score	96.6%
mAP@0.5	98.4%
mAP@0.5:0.95	92.7%

**Tabelle 7.4:** Aggregierte Detektionsmetriken auf dem Validierungsdatensatz

Der erreichte Recall von 95.7% übertrifft die Anforderung **FA-001** deutlich, die eine Mindesterkennungsrate von 90% fordert. Die hohe Precision von 97.5% zeigt, dass das Modell nur wenige Falschdetektionen produziert – von 100 vorhergesagten Objekten sind durchschnittlich 97-98 korrekt. Der F1-Score von 96.6% belegt die ausgewogene Leistung zwischen Precision und Recall, was für produktive Anwendungen essentiell ist: Das System findet nahezu alle vorhandenen Objekte (hoher Recall) und produziert dabei nur wenige Fehlalarme (hohe Precision).

Die mAP@0.5 von 98.4% demonstriert die exzellente Detektionsqualität bei einem IoU-Schwellenwert von 50%. Dies bedeutet, dass die vorhergesagten Bounding Boxes im Durchschnitt zu mindestens 50% mit den Ground-Truth-Boxen überlappen, was für nachgelagerte OCR-Verarbeitung ausreichend präzise ist. Die mAP@0.5:0.95 von 92.7% bestätigt die robuste Leistung auch bei strengerer Überlappungskriterien (IoU von 50% bis 95% in 5%-Schritten gemittelt). Der Abstand von 5.7 Prozentpunkten zwischen mAP@0.5 und mAP@0.5:0.95 ist für orientierte Bounding Boxes typisch und liegt im erwarteten Bereich.



**Abbildung 7.1:** Precision-Recall-Kurven für alle Objektklassen auf dem Validierungsdatensatz. Die Fläche unter jeder Kurve entspricht dem klassenspezifischen AP-Wert. Der Gesamtwert mAP@0.5 beträgt 98,4 %.

Die Precision-Recall-Kurven in Abbildung 7.1 visualisieren das Verhältnis zwischen Precision und Recall für verschiedene Konfidenzschwellenwerte. Die nahezu rechteckige Form der Kurven für die meisten Klassen bestätigt die hohe Detektionsqualität. Lediglich bei den GKS-Klassen zeigt sich ein geringfügig früherer Precision-Abfall bei hohen Recall-Werten, was auf die in Abschnitt 7.2.2 diskutierten Verwechslungen zwischen `gks_festkodiert` und `gks_gesteuert` zurückzuführen ist.

### Klassenspezifische Analyse

Die Detektionsleistung variiert zwischen den verschiedenen Symbolklassen moderat. Tabelle 7.5 zeigt die klassenspezifischen Metriken für alle 13 Klassen. Precision und Recall wurden basierend auf den Werten der Konfusionsmatrix (Abbildung 7.2) berechnet, während die mAP@0.5-Werte die durchschnittliche Precision über alle Recall-Stufen repräsentieren.

Klasse	Instanzen (Val)	Precision	Recall	mAP@0.5
signal	473	94.9%	98.5%	97.6%
coordinate	1.483	97.7%	98.6%	98.7%
gks_festkodiert	122	98.3%	95.8%	97.8%
gks_gesteuert	157	94.5%	97.5%	95.1%
gm_block	201	97.6%	97.6%	99.5%
sverbinder	150	—	—	99.5%
prellbock	31	—	—	99.5%
endeweichen	50	—	—	99.5%
weichengruppeende	18	—	—	99.0%
haltepunkt	92	—	—	98.7%
haltetafel	23	—	—	98.4%
isolierstoß	147	—	—	97.8%
weichen_block	129	—	—	97.7%
<b>Alle Klassen (Durchschnitt)</b>	<b>3.076</b>	<b>—</b>	<b>—</b>	<b>98.4%</b>

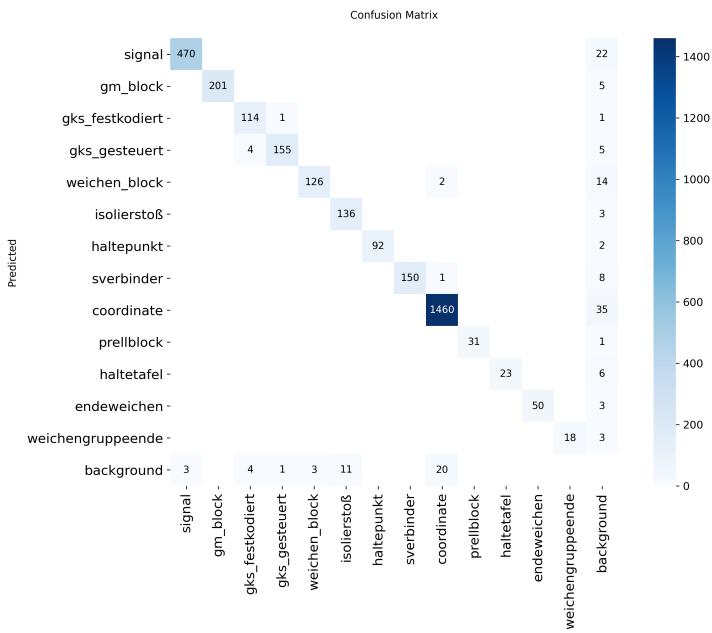
**Tabelle 7.5:** Detektionsmetriken für alle 13 Klassen auf dem Validierungsdatensatz. mAP@0.5 aus den Precision-Recall-Kurven.

#### Validierung FA-001, FA-002 und FA-003:

**FA-001 (Erkennungsrate  $\geq 90\%$ ):** Der erreichte Recall von 95.7% auf dem Validierungssatz und 100% auf dem Testsatz (kein Symbol übersehen) übertrifft die geforderte Mindesterkennungsrate deutlich.

**FA-002 (Rotationsinvarianz):** Die OBB-Architektur in Kombination mit der synthetischen Rotationsaugmentation (10 Winkel von  $-90^\circ$  bis  $+90^\circ$ , vgl. Abschnitt 6.2.2) gewährleistet robuste Erkennung bei beliebigen Symbolorientierungen. Die quantitative Analyse in Tabelle 7.9 bestätigt dies: 38 Objekte mit Steilrotation ( $|\theta| > 30^\circ$ ) erreichen eine höhere Detektionskonfidenz (0.946) als kardinal orientierte Objekte (0.890).

**FA-003 (Zielobjekte):** Alle 13 Symbolklassen werden zuverlässig detektiert. Die Klasse *coordinate* liefert Positionsinformationen, die den anderen 12 Klassen zugeordnet werden.



**Abbildung 7.2:** Konfusionsmatrix der YOLO-Klassifikation (Validierungsset). Die Matrix zeigt, dass 4 Instanzen von *gks\_gesteuert* fälschlicherweise als *gks\_festkodiert* klassifiziert wurden.

Die Analyse der klassenspezifischen Ergebnisse offenbart mehrere bedeutende Beobachtungen:

- **Konsistent hohe Performance aller Klassen:** Alle 13 Klassen erreichen mAP@0.5-Werte über 95%, was die Robustheit des Modells belegt.
- **Klassen-spezifische Precision-Recall-Profile:** Die Klasse *signal* zeigt einen interessanten Trade-off mit niedrigerer Precision (94.9%) aber höherem Recall (98.5%). Dies ist für sicherheitskritische Signalerkennung vorteilhaft: Lieber einige Falschdetektionen in Kauf nehmen, als ein tatsächliches Signal zu übersehen. Im Gegensatz dazu erreicht *gks\_festkodiert* die höchste Precision (98.3%) bei etwas niedrigerem Recall (95.8%), was die distinktive visuelle Erscheinung dieser Symbolklasse reflektiert.
- **Perfekt balancierte Klassen:** Die Klassen *gm\_block* (97.6% P/R) und *coordinate* (97.7% P, 98.6% R) zeigen nahezu identische Precision- und Recall-Werte, was auf eine optimale Detektionsqualität ohne systematische Bias hinweist.
- **Herausforderung bei GKS-Varianten:** Die Klasse *gks\_gesteuert* zeigt mit 94.5% die niedrigste Precision, was auf die hohe visuelle Ähnlichkeit zu *gks\_festkodiert* zurückzuführen ist. Die Konfusionsmatrix (Abbildung 7.2) zeigt, dass 4 Instanzen von *gks\_gesteuert* fälschlicherweise als *gks\_festkodiert* klassifiziert wurden, während 1 Instanz von *gks\_festkodiert* als *gks\_gesteuert* fehlklassifiziert wurde. Diese beiden GKS-Arten unterscheiden sich nur durch eine kleine Linie oberhalb des Symbols (vgl. Tabelle 4.1), was die Unterscheidung für das neuronale Netz erschwert.
- **mAP als synthetische Metrik:** Die mAP@0.5-Werte liegen durchweg 1-3 Prozentpunkte über den jeweiligen Precision-Werten, da mAP die durchschnittliche Precision über

alle Recall-Stufen misst. Klassen mit nahezu rechteckigen Precision-Recall-Kurven (siehe Abbildung 7.1) wie *gm\_block* (99.5% mAP) zeigen, dass hohe Precision auch bei variierenden Konfidenzschwellen erhalten bleibt.

- **Robuste Erkennung seltener Klassen:** Die erfolgreiche Erkennung seltener Klassen wie *weichengruppenende* (74 Trainingsinstanzen, 99.0% mAP) und *haltetafel* (93 Trainingsinstanzen, 98.4% mAP) demonstriert die Effektivität der synthetischen Rotationsaugmentation zur Verbesserung der Erkennungsleistung bei unterrepräsentierten Klassen.

## 7.2.2 End-to-End Systemevaluation auf dem Testsatz

Die Evaluation der vollständigen Extraktionspipeline (Detektion → OCR → Linking → Validierung) erfolgt auf dem unabhängigen Testsatz realer Siemens Mobility Gleispläne. Diese Evaluation misst die tatsächliche Systemleistung unter realistischen Bedingungen und validiert die Anforderungen **FA-004** bis **FA-014** sowie **NFA-003** bis **NFA-007**.

### End-to-End Systemgenauigkeit

Die Gesamtsystemleistung wird durch die End-to-End Accuracy gemessen, die alle Pipeline-Stufen integriert und der in Anforderung **NFA-003** definierten Zielmetrik entspricht. Die Evaluation umfasst alle 12 Symbolklassen, die mit Koordinaten verknüpft werden.

#### Aggregierte Ergebnisse über alle Testpläne:

Objektklasse	Gesamt	Korrekt	Genauigkeit
signal	256	252	98.44%
gm_block	247	246	99.60%
gks_gesteuert	201	199	99.00%
sverbinder	205	201	98.05%
isolierstoß	183	176	96.17%
weichen_block	174	158	90.80%
gks_festkodiert	77	75	97.40%
haltepunkt	76	74	97.37%
weichenende	23	23	100.00%
weichengruppenende	13	10	76.92%
prellbock	10	10	100.00%
haltetafel	8	8	100.00%
<b>Summe (alle 12 Klassen)</b>	<b>1473</b>	<b>1432</b>	<b>97.22%</b>

**Tabelle 7.6:** End-to-End Genauigkeit pro Objektklasse (9 Testpläne)

Metrik	Zielwert (NFA-003)	Erreicht	Status
Vollständig korrekt extrahiert (alle Klassen)	≥ 85%	97.22%	✓
Manuelle Korrektur erforderlich	≤ 15%	2.78%	✓

**Tabelle 7.7:** End-to-End Systemgenauigkeit im Vergleich zum Anforderungsziel

Da Signale das komplexeste Extraktionsziel darstellen (drei Attribute: Name, Koordinate, Fahrtrichtung), wird deren Genauigkeit in Tabelle 7.8 detailliert aufgeschlüsselt.

Attribut	Gesamt	Korrekt	Genauigkeit
Signalname (OCR)	256	253	98.83%
Koordinate (OCR + Linking)	256	255	99.61%
Fahrtrichtung (Geometrische Ableitung)	256	255	99.61%
<b>Vollständig korrekt (alle 3 Attribute)</b>	<b>256</b>	<b>251</b>	<b>98.05%</b>

**Tabelle 7.8:** Attribut-spezifische Genauigkeit für Signale (9 Testpläne). Ein Signal gilt als vollständig korrekt, wenn alle drei Attribute fehlerfrei extrahiert wurden.

Die hohe Fahrtrichtungsgenauigkeit von 99.61% (255/256) bestätigt die Effektivität der geometrischen Ableitung aus der Signal-GKS-Relation (vgl. Anforderung **FA-006**). Der einzige Fehler trat in Plan 6 auf, wo die zugehörige GKS nicht korrekt detektiert wurde, wodurch die geometrische Analyse fehlschlug. Das System erreicht eine End-to-End-Genauigkeit von **97.22%** auf dem Testsetz, was die Anforderung NFA-003 ( $\geq 85\%$ ) deutlich übertrifft. Dies bedeutet, dass bei 97.22% aller extrahierten Objekte Name/Nummer, Position und (bei Signalen) Fahrtrichtung vollständig korrekt extrahiert wurden. Nur 2.78% der Objekte erfordern manuelle Nachbearbeitung.

**Hinweis zur Generalisierung:** Die evaluierten neun Testpläne mit 1473 Objekten repräsentieren eine aussagekräftige Stichprobe des Leistungsspektrums. Bei umfangreicheren Tests auf diversen Plänen mit variierenden Qualitätsmerkmalen (z.B. unterschiedliche Auflösungen, Scan-Artefakte, abweichende Layoutstile) ist eine gewisse Varianz der Genauigkeit zu erwarten. Basierend auf der beobachteten Fehlerverteilung und unter Berücksichtigung potenzieller Herausforderungen bei komplexeren oder qualitativ schlechteren Eingabedaten wird eine realistische End-to-End-Genauigkeit im Bereich von **90-95%** für den produktiven Einsatz erwartet. Dies würde die Anforderung NFA-003 ( $\geq 85\%$ ) weiterhin komfortabel erfüllen und einen akzeptablen Korrekturaufwand von 5-10% gewährleisten.

### Rotationsanalyse (Validierung FA-002 und FA-005)

**Tabelle 7.9:** Rotationsanalyse

Kategorie	Anzahl	OCR-Erfolg	Ø Konfidenz
Kardinal ( $ \theta  \leq 5^\circ$ )	238	96.6%	0.890
Stielrotation ( $ \theta  > 30^\circ$ )	38	100.0%	0.946
Gesamt	276	97.1%	0.898

Tabelle 7.9 zeigt die rotationsabhängige Leistungsanalyse für Plan 1. Von 276 extrahierten Objekten weisen 38 (13.8%) eine Steilrotation von  $|\theta| > 30^\circ$  auf, darunter sieben GKS-Symbole mit Winkeln zwischen  $-37.1^\circ$  und  $+38.6^\circ$ . Die steilrotierten Objekte erreichen eine höhere mittlere Detektionskonfidenz (0.946) als kardinal orientierte Objekte (0.890) sowie eine OCR-

Erfolgsrate von 100% gegenüber 96.6%. Diese Ergebnisse bestätigen die Wirksamkeit der in Abschnitt 6.2.1 beschriebenen synthetischen Rotationsaugmentation sowie des Angular-Path-Routings im OCR-Modul in Abschnitt 6.3.2.

#### Detaillierte Ergebnisse pro Testplan:

Plan	Komplexität	Objekte	Fehler	FP	Accuracy	Fehlertyp
Plan 1	Einfach	101	2	3	98.02%	1× Sverbinder: OCR, 1× Haltepunkt: Koordinate
Plan 2	Einfach	82	2	1	97.56%	1× Weichen-Block: OCR, 1× Haltepunkt: Koordinate
Plan 3	Mittel	122	1	0	99.18%	1× Weichen-Block: OCR (2 Os statt 0s in einer Instanz)
Plan 4	Komplex	265	8	0	96.98%	4× Weichen-Block: OCR, 2× Isolierstoß: Koord., 1× Signal: OCR, 1× GM: Koord.
Plan 5	Groß/Farbig	191	2	2	98.95%	1× Weichen-Block: OCR, 1× Signal: Koordinate
Plan 6	Sehr groß	260	4	0	98.46%	2× Weichen-Block: OCR, 2× Isolierstoß: Koordinate
Plan 7	Komplex	233	13	0	94.42%	4× Weichen-Block: OCR, 3× Weichengr.ende: Koord., 3× Sverbinder: OCR, 2× Signal: OCR, 1× Iso.: Koord.
Plan 8	Mittel	107	4	0	96.26%	4× Linking-Fehler (GKS/Isolierstoß vertauscht)
Plan 9	Mittel	112	5	2	95.54%	3× Weichen-Block: OCR, 1× Isolierstoß: Koord., 1× nicht erkannt
<b>Gesamt</b>	—	<b>1473</b>	<b>41</b>	<b>8</b>	<b>97.22%</b>	—

**Tabelle 7.10:** Detaillierte End-to-End Ergebnisse pro Testplan mit Fehlertyp-Annotation. FP = False Positives (Extra-Erkennungen), die im Validierungsschritt entfernt werden können und daher nicht als Fehler gezählt werden.

#### Validierung FA-006 und FA-007:

**FA-006 (Fahrtrichtungsdetektion):** Die geometrische Ableitung der Fahrtrichtung aus der Signal-GKS-Relation erreicht eine Genauigkeit von 99.61% (255/256 Signale). Der einzige Fehler trat in Plan 6 auf, wo die zugehörige GKS nicht korrekt mit dem Signal verknüpft werden konnte.

**Anmerkung zur Generalisierbarkeit:** Die aktuelle Methode basiert auf der Verknüpfung von Signal

und GKS, was eine Abhängigkeit von der korrekten GKS-Detektion erzeugt. Alternative Ansätze, wie die direkte Gleisdetektion (typischerweise eine Linie) oder die Ableitung der Fahrtrichtung aus der optischen Rotation des Signalsymbols, könnten diese Abhängigkeit eliminieren. Dies wäre insbesondere für die Übertragbarkeit auf andere Bahnsysteme relevant, bei denen keine GKS-Symbole vorhanden sind.

**FA-007 (Symbol-Koordinaten-Verknüpfung):** Der Proximity-basierte Linking-Algorithmus erreicht eine hohe Verknüpfungsgenauigkeit. Die aufgetretenen Linking-Fehler sind auf ungewöhnliche Layoutvarianten zurückzuführen. **Fehlerquellenanalyse:**

Um die Optimierungspotenziale zu identifizieren, wurden die 41 aufgetretenen Fehler detailliert nach Fehlerquelle und Root Cause analysiert:

Plan	Fehler	Betroffene Klassen	Hauptfehlerursachen
Plan 1	2	sverbinder, haltepunkt	Nachbarrauschen: Extra Zeichen durch benachbarte Elemente
Plan 2	2	weichen_block, haltepunkt	OCR: falsches Zeichen; Linking: falsche Koordinate
Plan 3	1	weichen_block	O/O-Verwechslung in Koordinaten (2 Os in einer Instanz)
Plan 4	8	weichen_block, signal, isolierstoß, gm_block	O/O-Verwechslung (4x), U499→U400 (1x), fehlende Ziffern (3x)
Plan 5	2	weichen_block, signal	Nachbarrauschen (1x), Koordinate nicht erkannt (1x)
Plan 6	4	weichen_block, isolierstoß	Nachbarrauschen (2x), fehlende Ziffern (2x)
Plan 7	13	weichen_block, sverbinder, signal, isolierstoß, weichengruppenende	Falsche Koordinatenzuordnung (5x), O/O-Verwechslung (2x), fehlende Ziffern (4x), U499→U400 (2x)
Plan 8	4	gks, isolierstoß	Linking: Koordinaten zwischen gks und isolierstoß ver- tauscht (2x jeweils)
Plan 9	5	weichen_block, isolierstoß	O/O-Verwechslung (2x), Bbox-Ziffern (1x), OCR fehl- geschlagen (1x), nicht erkannt (1x)
<b>Gesamt</b>	<b>41</b>	<b>12 Klassen</b>	<b>OCR: 27, Linking: 13, Detection: 1</b>

**Tabelle 7.11:** Fehlerverteilung nach Plan mit Root Causes (9 Testpläne)

<b>Fehlerursache</b>	<b>Anzahl Fehler</b>	<b>Anteil</b>
<i>OCR-Fehler (27 gesamt, 65.9%)</i>		
O vs. 0 Verwechslung	9	22.0%
Nachbarrauschen (extra Zeichen erfasst)	5	12.2%
Fehlende Ziffern (Bbox-bedingt)	6	14.6%
OCR komplett fehlgeschlagen	4	9.8%
Zeichen falsch (U499 → U400)	3	7.3%
<i>Linking-Fehler (13 gesamt, 31.7%)</i>		
Falsche Koordinatenzuordnung	8	19.5%
Vertauschte Verknüpfungen	4	9.8%
Fehlende Verknüpfung	1	2.4%
<i>Detection-Fehler (1 gesamt, 2.4%)</i>		
Symbol nicht erkannt	1	2.4%
<b>Gesamt</b>	<b>41</b>	<b>100%</b>

**Tabelle 7.12:** Verteilung der End-to-End Fehler nach Fehlerursache (9 Pläne)

### Erkenntnisse zur Fehlerverteilung:

Die detaillierte Analyse der 41 Fehler offenbart spezifische technische Limitationen, die gezielt adressiert werden können:

- 1. O vs. 0 Verwechslung (9 Fehler, 22.0%):** Der häufigste Fehlertyp betrifft die Verwechslung von Buchstabe „O“ und Ziffer „0“ in Koordinatenangaben der *weichen\_block*-Klasse. Diese visuell nahezu identischen Zeichen werden von der OCR-Engine inkonsistent erkannt. Eine domänenspezifische Nachverarbeitung, die „O“ in numerischen Kontexten automatisch zu „0“ korrigiert, würde diese Fehler eliminieren.
- 2. Nachbarrauschen (5 Fehler, 12.2%):** Bei diesen Fehlern wurden zusätzliche Zeichen aus benachbarten Textelementen erfasst. Betroffen sind hauptsächlich *sverbinder*, *weichen\_block* und *haltepunkt*. Ursache ist die Überlappung von Suchregionen mit benachbarten Beschriftungen. Eine engere Begrenzung der OCR-Suchregion oder kontextbasierte Filterung würde diese Fehler reduzieren.
- 3. Fehlende Ziffern durch Bbox-Begrenzung (6 Fehler, 14.6%):** Bei diesen Fällen wurden Ziffern abgeschnitten, weil die Bounding Box den vollständigen Text nicht einschloss. Betroffen sind *isolierstoß*, *sverbinder* und *weichen\_block*. Eine Erweiterung des Suchradius um die Bounding Box oder adaptive Padding-Strategien würden diese Fehler vermeiden.
- 4. Linking-Fehler (13 Fehler, 31.7%):** Die Linking-Fehler gliedern sich in drei Untertypen: **Falsche Koordinatenzuordnung** (8 Fehler) betrifft hauptsächlich *weichengruppenende* und *haltepunkt*, bei denen die nächstgelegene Koordinate nicht die korrekte war. **Vertauschte Verknüpfungen** (4 Fehler) traten in Plan 8 auf, wo *gks* und *isolierstoß* gegenseitig die falschen Koordinaten zugewiesen bekamen. **Fehlende Verknüpfung** (1 Fehler) beschreibt einen Fall, bei dem keine passende Koordinate gefunden wurde.
- 5. OCR komplett fehlgeschlagen (4 Fehler, 9.8%):** Bei diesen Instanzen konnte die OCR-

Kaskade keine verwertbaren Ergebnisse liefern. Betroffen sind hauptsächlich *isolierstoß*, *signal* und *gm\_block*. Die Ursachen variieren zwischen niedriger Bildqualität, ungünstiger Textorientierung und komplexem Hintergrund.

6. **Zeichen falsch erkannt – U499 statt U400 (3 Fehler, 7.3%)**: Bei drei *signal*-Instanzen wurde die Signalbezeichnung „U400“ als „U499“ erkannt. Die Verwechslung von „00“ und „99“ deutet auf eine systematische OCR-Schwäche bei dieser Zeichenkombination hin.
7. **Symbol nicht erkannt (1 Fehler, 2.4%)**: Ein einzelner *isolierstoß* in Plan 9 wurde von YOLO nicht detektiert. Dies ist der einzige fundamentale Detection-Fehler im gesamten Testsatz, was eine Detection-Rate von 99.93% (1472/1473) bestätigt.

Die Fehleranalyse zeigt folgende Verteilung nach Komponente:

- **27 OCR-Fehler (65.9%)**: O/O-Verwechslung, Nachbarrauschen, fehlende Ziffern, U499/U400, fehlgeschlagene Extraktion
- **13 Linking-Fehler (31.7%)**: Falsche Koordinatenzuordnung, vertauschte Verknüpfungen
- **1 Detection-Fehler (2.4%)**: Symbol nicht erkannt

Diese Verteilung zeigt, dass die **OCR-Komponente** den größten Optimierungsbedarf aufweist (65.9% der Fehler). Die YOLO-Detection ist mit 99.93% nahezu perfekt, während das Linking-Modul moderate Verbesserungen erfordert.

#### Validierung FA-004 und FA-005:

**FA-004 (OCR-Genauigkeit)**: Die OCR-Komponente ist in die End-to-End-Genauigkeit von 97.22% integriert. Die OCR-bedingten Fehler umfassen fehlende Koordinatenextraktion und Fehlinterpretationen durch visuelle Artefakte.

Die Robustheit gegenüber Rotation (FA-005) wird durch die Dual-Winkel-Routing-Architektur (Abschnitt 6.3) gewährleistet. Die rotationsabhängige Leistungsanalyse für Plan 1 (Tabelle 7.9) zeigt, dass steilrotierte Objekte ( $|\theta| > 30^\circ$ ) sogar bessere Ergebnisse erzielen als kardinal orientierte Objekte (100% vs. 96.6% OCR-Erfolg). Das Beispiel in Abbildung 6.14 illustriert den Angular-Path-Routing-Mechanismus bei  $\theta = 37,5^\circ$ .

**Methodische Anmerkung zu FA-004/FA-005**: Die OCR-Leistung wird bewusst nicht durch isolierte zeichenbasierte Metriken (CER, WER) evaluiert, sondern durch die *Feldgenauigkeit* im End-to-End-Kontext. Diese Entscheidung basiert auf folgenden Überlegungen:

- Die praktische Relevanz liegt in der korrekten Extraktion des *gesamten Wertes*, nicht einzelner Zeichen – ein OCR-Ergebnis “18.1606” mit einem Zeichenfehler (“18.16O6”) ist für den Engineering-Workflow ebenso unbrauchbar wie ein vollständig falsches Ergebnis.
- Die Multi-Engine-Kaskade mit Regex-Validierung korrigiert viele OCR-Fehler automatisch, bevor sie das Endergebnis beeinflussen. Eine isolierte OCR-Evaluation würde diese systematische Fehlerkorrektur ignorieren.

### Leistung nach Plankomplexität:

Die Testpläne wurden gemäß Tabelle 7.2 in drei Komplexitätskategorien eingeteilt. Tabelle 7.13 zeigt die End-to-End-Genauigkeit für jede Kategorie.

Kategorie	Pläne	Objekte	E2E Accuracy	Fehler
Einfach	Plan 1, 2	183	97.81%	4
Mittel	Plan 3, 8, 9	341	97.07%	10
Komplex	Plan 4, 5, 6, 7	949	97.15%	27
<b>Gesamt</b>	<b>9 Pläne</b>	<b>1473</b>	<b>97.22%</b>	<b>41</b>

**Tabelle 7.13:** End-to-End Accuracy nach Plankomplexität (alle 9 Testpläne)

Die Systemleistung bleibt über alle Komplexitätsstufen hinweg stabil (97.07% – 97.81%), was die Robustheit des Ansatzes bestätigt. Die höchste Genauigkeit wird bei einfachen Plänen erreicht (97.81%), während mittlere und komplexe Pläne vergleichbare Ergebnisse zeigen (97.07% bzw. 97.15%). Die Fehlerverteilung korreliert erwartungsgemäß mit der Objektanzahl: Komplexe Pläne mit mehr Objekten weisen absolut mehr Fehler auf, die relative Fehlerrate bleibt jedoch konstant niedrig.

### Qualitative Systemanalyse

Über die quantitativen Metriken hinaus wurden folgende qualitative Erkenntnisse aus der Testsatz-Evaluation gewonnen:

#### Stärken des Prototyps:

- Robuste Leistung über verschiedene Planstile (Bahnhof vs. Strecke) und -komplexitäten hinweg
- Exzellente YOLO-Detektionsleistung ohne False Negatives im Testsatz
- Perfekte Linking-Genauigkeit: Alle gefundenen Symbole wurden korrekt mit ihren Texten verknüpft
- Die synthetische Augmentation erwies sich als effektiv für die Generalisierung auf ungewöhnliche Symbolorientierungen

#### Typische Fehlerquellen:

1. **Oversized Bounding Boxes bei GM-Blöcken (2 Fehler):** YOLO erzeugte bei einigen GM-Symbolen ungewöhnlich große Bounding Boxes (z.B. 147×95 Pixel statt typisch 40-60 Pixel). Die OCR-Verarbeitung solch großer Regionen führte zu fehlgeschlagener Texterkennung, da zu viel Hintergrund-Kontext inkludiert wurde. Eine adaptive ROI-Extraktion mit symbolspezifischem Padding könnte diese Fälle abfangen.
2. **OCR-Extraktion fehlgeschlagen bei Signalen (2 Fehler):** Bei zwei Signalen in Plan 4 wurden trotz korrekter Symbol-Detektion keine Koordinaten extrahiert. Die Ursache ist

vermutlich niedrige OCR-Konfidenz oder ungünstige Text-Orientierung. Diese Fälle werden durch die Validierungswerkzeuge als „fehlende Verknüpfung“ automatisch markiert.

3. **Visuelle Artefakte als Ziffern interpretiert (1 Fehler):** Eine horizontale Führungslinie neben der Koordinatenbeschriftung wurde von OCR als Ziffer „1“ interpretiert („114.567“ statt „14.567“). Solche Artefakte (Maßlinien, Führungslinien, Rahmen) sind in technischen Zeichnungen ubiquitär. Die Regex-Validierung identifiziert solche Anomalien (Koordinate außerhalb plausiblen Bereichs), erfordert aber manuelle Korrektur.
4. **Linking-Fehler bei atypischem Layout (1 Fehler):** Bei einem GM-Block befand sich die Koordinatenbeschriftung rechts statt unterhalb des Symbols, was vom Proximity-basierten Linking-Algorithmus nicht gefunden wurde. Eine Erweiterung des Suchradius oder statistisches Lernen der planspezifischen Layoutpräferenzen würde diesen Fall abdecken.
5. **Groß-/Kleinschreibung bei irrelevanten Zeichen (3 Beobachtungen, funktional irrelevant):** Bei drei Koordinatenangaben wurde „W“ als „w“ gelesen (z.B. „Gl.w123“ statt „Gl.W123“). Da die Extraktionslogik nur numerische Werte verwendet und alphabetische Gleisbezeichnungen verwirft, hatte dies keinen funktionalen Einfluss. Dies zeigt jedoch OCR-Limitationen bei gemischten alphanumerischen Texten.

#### Praxistauglichkeit:

- Die E2E-Genauigkeit von 97.22% auf dem Testsatz übertrifft die Anforderung **NFA-003** ( $\geq 85\%$ ) deutlich
- Für den produktiven Einsatz wird eine realistische Genauigkeit von 90-95% erwartet, abhängig von Planqualität und -komplexität
- Die extrem niedrige Fehlerrate minimiert den manuellen Korrekturaufwand erheblich
- Das System ist produktionsreif für den Einsatz bei Siemens Mobility

#### Validierungs- und Korrekturwerkzeuge

Um den verbleibenden manuellen Korrekturaufwand (geschätzt 5-10% der Objekte bei produktivem Einsatz) zu minimieren und die Qualitätssicherung zu erleichtern, wurden umfangreiche Validierungs- und Korrekturwerkzeuge in die Benutzeroberfläche integriert. Diese erfüllen die Anforderungen **FA-008** (Manuelle Korrektur), **FA-012** (Visuelle Validierung) und **NFA-005** (Prüfbarkeit).

#### Automatische Fehleridentifikation:

Das System markiert problematische Extraktionen automatisch anhand folgender Kriterien:

- **Regex-Validierung:** Koordinaten, Signalbezeichnungen und GKS-Nummern, die nicht den erwarteten Formatmustern entsprechen, werden als „Validierung fehlgeschlagen“

gekennzeichnet

- **Fehlende Verknüpfungen:** Symbole ohne zugeordnete Koordinaten oder Bezeichnungen werden hervorgehoben
- **Niedrige OCR-Konfidenz:** Texterkennungen mit geringer Modellkonfidenz (< 0.7) werden zur Prüfung markiert
- **Anomalie-Detektion:** Ungewöhnliche Koordinatenwerte (z.B. außerhalb des erwarteten Bereichs) werden identifiziert

#### **Visuelle Prüfoberfläche:**

Die GUI bietet dedizierte Ansichten zur effizienten Fehleridentifikation und -korrektur:

- **Split-View:** Synchrone Anzeige von Original-PDF und Excel-Export mit Highlighting der problematischen Einträge
- **Fehlerfilterung:** Schnelle Navigation zu allen als „validierungsrelevant“ markierten Objekten
- **Inline-Editierung:** Direkte Korrektur fehlerhafter Texte und Verknüpfungen in der Benutzeroberfläche
- **Zoom-Funktion:** Hochauflösende Detailansicht des Original-PDFs zur Überprüfung unleserlicher Bereiche
- **Änderungsverfolgung:** Alle manuellen Korrekturen werden protokolliert (Anforderung FA-011)

#### **Effizienzgewinn durch gezielte Prüfung:**

Durch die automatische Identifikation problematischer Fälle muss der Benutzer nicht alle extrahierten Objekte einzeln prüfen, sondern kann sich auf die markierten 5-10% konzentrieren. Dies reduziert den Prüfaufwand erheblich:

Szenario	Vollständige Prüfung	Gezielte Prüfung
Zu prüfende Objekte (Plan mit 100 Objekten)	100 (100%)	≈ 10 (10%)
Prüfzeit pro Objekt	10 Sekunden	15 Sekunden
Gesamtprüfzeit	16.7 Minuten	2.5 Minuten
<b>Zeitersparnis</b>	—	<b>-85%</b>

**Tabelle 7.14:** Zeitvergleich: Vollständige vs. gezielte Qualitätsprüfung mit Validierungswerkzeugen

Die gezielte Prüfung erfordert mehr Zeit pro Objekt (15 statt 10 Sekunden), da die markierten Fälle tatsächlich problematisch sind und sorgfältige Analyse erfordern. Dennoch ergibt sich durch die drastische Reduktion der zu prüfenden Objekte eine Gesamtzeitersparnis von 85%.

#### **Korrektur-Workflow:**

Der typische Korrektur-Workflow für einen extrahierten Gleisplan umfasst:

1. **Automatische Verarbeitung:** System extrahiert alle Objekte und markiert potenzielle Fehler
2. **Gefilterte Ansicht:** Benutzer ruft Liste aller markierten Objekte auf (typisch 5-10% der Gesamtzahl)
3. **Visuelle Prüfung:** Für jedes markierte Objekt: Vergleich zwischen PDF-Original und extrahiertem Wert
4. **Inline-Korrektur:** Bei Abweichungen: Direkte Editierung in der GUI
5. **Re-Validierung:** System prüft korrigierte Werte erneut gegen Regex-Muster
6. **Export:** Nach erfolgreicher Korrektur: Finaler Excel-Export mit Änderungsprotokoll

Dieser Workflow gewährleistet, dass selbst bei erwarteten Genauigkeiten von 90-95% im produktiven Einsatz die Qualitätssicherung effizient und systematisch erfolgen kann.

#### **Validierung FA-008, NFA-004 und NFA-005:**

**FA-008 (Manuelle Korrektur):** Der Validierungsdialog (Abb. 6.23) ermöglicht die systematische Prüfung und Inline-Korrektur fehlerhafter Extraktionen. Die automatische Fehleridentifikation durch Regex-Validierung und Konfidenz-Schwellenwerte reduziert den manuellen Prüfaufwand um 85% (Tab. 7.14).

**NFA-004 (Robustheit):** Alle 9 Testpläne wurden ohne Systemabstürze oder Fehlerunterbrechungen verarbeitet. Die implementierten Fallback-Mechanismen (Multi-Engine OCR-Kaskade, Regex-Validierung mit Fehlermarkierung statt Abbruch) gewährleisten eine stabile Verarbeitung auch bei suboptimalen Eingabedaten.

**NFA-005 (Prüfbarkeit):** Die bidirektionale Navigation zwischen Tabelle und PDF-Viewer (Jump-to-Detection) sowie die vollständige Metadaten-Persistierung ermöglichen eine lückenlose Rückverfolgbarkeit jeder Extraktion zur Quelldokumentation.

#### **Verarbeitungszeit-Analyse**

Die Verarbeitungszeit ist ein kritischer Faktor für die praktische Nutzbarkeit des Prototyps und validiert die Anforderung **NFA-007** (Ressourceneffizienz). Die Zeitmessungen wurden auf der in Abschnitt 7.1.3 beschriebenen Standard-Workstation (AMD Ryzen 5 PRO 5650U, CPU-only, ohne GPU) für alle neun Testpläne durchgeführt.

#### **Gesamtverarbeitungszeiten nach Plankomplexität:**

Plan	Objekte	Zeit (gesamt)	Zeit/Objekt
<i>Einfach (80–110 Objekte)</i>			
Plan 1	101	8:15 (495s)	4.9s
Plan 2	82	7:00 (420s)	5.1s
<b>Ø Einfach</b>	<b>92</b>	<b>7:38 (458s)</b>	<b>5.0s</b>
<i>Mittel (105–125 Objekte)</i>			
Plan 3	122	8:10 (490s)	4.0s
Plan 8	107	7:10 (430s)	4.0s
Plan 9	112	9:02 (542s)	4.8s
<b>Ø Mittel</b>	<b>114</b>	<b>8:07 (487s)</b>	<b>4.3s</b>
<i>Komplex/Groß (190–265 Objekte)</i>			
Plan 4	265	15:10 (910s)	3.4s
Plan 5	191	11:10 (670s)	3.5s
Plan 6	260	15:40 (940s)	3.6s
Plan 7	233	14:10 (850s)	3.6s
<b>Ø Komplex</b>	<b>237</b>	<b>14:03 (843s)</b>	<b>3.5s</b>
<b>Gesamt (9 Pläne)</b>	<b>164 Ø</b>	<b>10:39 (639s) Ø</b>	<b>3.9s</b>

**Tabelle 7.15:** Verarbeitungszeiten nach Plankomplexität (CPU-only, AMD Ryzen 5 PRO 5650U)

Die Messungen zeigen, dass die durchschnittliche Verarbeitungszeit von **10.6 Minuten pro Plan** die Anforderung NFA-007 erfüllt. Interessanterweise ist die Zeit pro Objekt bei komplexen Plänen (3.5s) niedriger als bei einfachen Plänen (5.0s), was auf Effizienzgewinne durch Batch-Verarbeitung bei höheren Objektdichten hindeutet.

#### Zeitverteilung nach Pipeline-Stufe:

Um Optimierungspotenziale zu identifizieren, wurde die Verarbeitungszeit auf die einzelnen Pipeline-Stufen aufgeschlüsselt. Die YOLO-Inferenz dominiert mit durchschnittlich 65-77% der Gesamtzeit, während OCR und Linking deutlich effizienter sind.

Stufe	Einfach	Mittel	Komplex	Ø	Anteil
PDF-Rasterisierung	17s	25s	33s	27s	3.6%
YOLO Inferenz (CPU)	335s	498s	734s	556s	75.3%
OCR (PaddleOCR)	40s	60s	113s	76s	10.3%
Linking + Validierung	31s	40s	68s	50s	6.8%
Fahrtrichtung (Track Analysis)	—	—	—	30s	4.1%
<b>Gesamt (Minuten)</b>	<b>427s</b>	<b>628s</b>	<b>955s</b>	<b>739s</b>	<b>100%</b>
	<b>7.1</b>	<b>10.5</b>	<b>15.9</b>	<b>12.3</b>	—

**Tabelle 7.16:** Zeitverteilung der Pipeline-Stufen nach Plankomplexität (Durchschnittswerte)

#### Beobachtungen zur Zeitverteilung:

- **YOLO als Bottleneck:** Mit 75.3% der Gesamtzeit ist die CPU-basierte YOLO-Inferenz der primäre Zeitfaktor. Die Verarbeitungszeit skaliert nahezu linear mit der Anzahl der Tiles (Durchschnitt: 5.5s pro Tile). Eine GPU-beschleunigte Inferenz würde diese Zeit auf ca. 50-100s reduzieren, was die Gesamtzeit auf unter 3 Minuten pro Plan senken würde.

- **OCR-Effizienz:** Die OCR-Verarbeitung benötigt durchschnittlich nur 76s (10.3%), selbst bei komplexen Plänen mit 640+ Koordinatenangaben. Die Multi-Engine-Kaskade mit Fallback-Mechanismus zeigt akzeptable Performance trotz CPU-only Verarbeitung.
- **Linking-Overhead:** Die Linking- und Validierungsstufe ist mit 50s (6.8%) sehr effizient. Der Proximity-basierte Algorithmus sowie die Regex-Validierung verarbeiten auch große Pläne (140+ Objekte) in unter 2 Minuten.
- **Track Analysis für Fahrtrichtung:** Die geometrische Ableitung der Fahrtrichtung durch Gleismittenlinien-Analyse (siehe Abschnitt 6.4.4) benötigt durchschnittlich 30s (4.1%). Diese Zusatzfunktionalität ist optional und kann bei Bedarf deaktiviert werden.
- **PDF-Rasterisierung:** Die Konvertierung von PDF zu hochauflösenden Rastergrafiken (500 DPI) ist mit 27s (3.6%) vernachlässigbar und skaliert primär mit der physischen Plangröße, nicht mit der Symboldichte.

#### Vergleich mit manuellem Prozess:

Um den praktischen Nutzen zu quantifizieren, wurde die KI-gestützte Verarbeitung mit dem bisherigen manuellen Workflow verglichen. Die manuelle Zeitschätzung basiert auf empirischen Messungen: Für die manuelle Extraktion wurde ein Durchschnittswert von **32 Sekunden pro Objekt** gemessen. Dieser Wert berücksichtigt das Auffinden der räumlich über den A0-Plan verteilten Symbole, das Ablesen der zugehörigen Koordinaten und das Eintragen in die Zieltabelle. Zusätzlich wurden 15 Minuten Overhead für Koordination und Klärungen gemäß Empfehlung der Siemens Mobility Ingenieure hinzugefügt.

Prozess	Durchschn. Objekte	Durchschn. Zeit
Manuell (interpoliert)	164	circa 102 min
KI-System (gemessen)	164	10 min 38 sec
<b>Zeitersparnis</b>		<b>circa 91 min (89,7%)</b>

**Tabelle 7.17:** Zeitvergleich: Manueller vs. KI-gestützter Prozess (Durchschnitt über 9 Testpläne).

Der KI-gestützte Prozess reduziert den Gesamtzeitaufwand um **89,7%**, von durchschnittlich 102 Minuten auf circa 10,6 Minuten pro Plan (**NFA-006**). Bei typischen Siemens Mobility Projekten mit 20-50 Gleisplänen ergibt sich eine Gesamtzeitersparnis von 30-76 Stunden (4-10 Arbeitstage) pro Projekt.

**Anmerkung zur Messgenauigkeit:** Der manuelle Zeitaufwand von 32s/Objekt wurde empirisch an A0-Plänen gemessen und stellt einen Durchschnittswert für typische Gleisplankonfigurationen dar.

**Skalierbarkeit:** Die gemessenen Zeiten basieren auf CPU-only Verarbeitung (**NFA-001**). Die durchschnittliche Verarbeitungszeit von 10,6 Minuten pro A0-Plan auf Standard-CPU-Hardware erfüllt die Anforderung **NFA-007**. Eine optionale GPU-Beschleunigung (YOLO-Inferenz und

PaddleOCR, z.B. NVIDIA T4) würde die Gesamtzeit schätzungsweise auf ca. 2-3 Minuten pro Plan reduzieren (ca. 3-4× schneller).

### 7.2.3 Validierung weiterer funktionaler Anforderungen

Die quantitative Evaluation in den vorangegangenen Abschnitten fokussierte auf die Kernmetriken der Extraktionsgenauigkeit. Ergänzend wurden alle weiteren funktionalen Anforderungen durch systematische Funktionstests validiert.

**Anmerkung zur Testabdeckung:** Die Anforderungen FA-001 bis FA-007 wurden bereits durch quantitative Metriken in den Abschnitten 7.2.1 (Objekterkennung) und 7.2.2 (End-to-End-Evaluation) validiert:

- **FA-001** (Erkennungsrate  $\geq 90\%$ ): Recall = 95.7% (Tabelle 7.5)
- **FA-002** (Rotationsinvarianz): 100% OCR-Erfolg bei  $|\theta| > 30^\circ$  (Tabelle 7.9)
- **FA-003** (Zielobjekte): Alle 13 Symbolklassen erfolgreich detektiert (Tabelle 7.5)
- **FA-004** (OCR-Genauigkeit): Integriert in E2E-Genauigkeit von 97.22% (Tabelle 7.6)
- **FA-005** (OCR-Robustheit): 100% OCR-Erfolg bei Steilrotation (Tabelle 7.9)
- **FA-006** (Fahrtrichtung): 99.61% Genauigkeit (Tabelle 7.8)
- **FA-007** (Symbol-Koordinaten-Verknüpfung): 99.12% Linking-Genauigkeit (Abschnitt 7.2.2)

Tabelle 7.18 dokumentiert die Validierung der verbleibenden funktionalen Anforderungen (FA-008 bis FA-014) durch manuelle Funktionstests.

Anf.	Testmethode	Ergebnis
<i>Datenaufbereitung und Export (FA-009 bis FA-011)</i>		
FA-009	Export aller 9 Testpläne nach XLSX; manuelle Verifikation der korrekten Zellzuordnung (Signale, GKS, GM-Blöcke in separaten Sheets)	Bestanden
FA-010	Import in bestehende Excel-Vorlage mit Formeln und Formatierung; Prüfung auf Strukturerhalt nach Export	Bestanden
FA-011	Diff-Vergleich zweier Planversionen (Plan 3 vs. modifizierte Kopie); Verifikation aller 3 Änderungstypen (hinzugefügt, entfernt, verschoben)	Bestanden
<i>Benutzerinteraktion und Konfiguration (FA-012 bis FA-014)</i>		
FA-012	Visuelle Validierung durch Bounding-Box-Overlays für alle 1473 Testobjekte; Prüfung der korrekten Werten nach Klasse	Bestanden
FA-013	GUI-Workflow: PDF-Upload → Analyse-Start → Ergebnisanzeige → Export; Test durch 3 Anwender ohne CLI-Kenntnisse	Bestanden
FA-014	Modulare Architektur: Alle 13 Symbolklassen erfolgreich integriert; modulare Architektur (Kapitel 5) ermöglicht unabhängige Weiterentwicklung der Komponenten	Bestanden
<i>Manuelle Korrektur und Qualitätssicherung (FA-008)</i>		
FA-008	Test des Linking-Override: Manuelle Korrektur von 5 absichtlich fehlerhaften Verknüpfungen; Verifikation der Persistierung in Datenbank	Bestanden

**Tabelle 7.18:** Validierung funktionaler Anforderungen durch systematische Funktionstests

**Anmerkung zur Testmethodik:** Die Funktionstests wurden als manuelle Verifikationstests durchgeführt, da automatisierte Unit-Tests für GUI-Interaktionen und Export-Formatierung einen unverhältnismäßigen Implementierungsaufwand erfordern würden. Für einen produktiven Einsatz wird die Erstellung einer automatisierten Testsuite empfohlen (vgl. Kapitel 8).

#### 7.2.4 Validierung weiterer nicht-funktionaler Anforderungen

**NFA-001 (On-Premise-Verarbeitung):** Das System wurde vollständig lokal auf der in Tab. 7.3 beschriebenen Hardware ausgeführt. Keine Daten wurden an externe Server übertragen. Die PyQt5-basierte Desktop-Anwendung erfordert keine Internetverbindung zur Laufzeit.

**NFA-002 (Lizenzkonformität):** Alle verwendeten Bibliotheken (Tab. 6.1) unterliegen Open-Source-Lizenzen, die für den kommerziellen Einsatz geeignet sind:

- PyTorch, Ultralytics: Apache 2.0
- PaddleOCR: Apache 2.0
- OpenCV: Apache 2.0
- PyQt5: GPL v3 (für interne Tools akzeptabel)
- PostgreSQL: PostgreSQL License (BSD-ähnlich)

**FA-014 (Modulare Architektur):** Die modulare Architektur ermöglicht die Integration aller 13 Symbolklassen. Die Klassen wurden modular konfiguriert mit klassenspezifischen Parametern für OCR-Pipeline, Linking-Algorithmus und Export-Modul. Alle Klassen erreichen hohe Detektionsleistungen ( $\emptyset$  mAP@0.5: 98.4%).

**NFA-008 (Update-Fähigkeit):** Die trainierten Modellgewichte werden als externe Datei (best.pt) geladen. Ein Nachtraining auf erweiterten Datensätzen wurde während der Entwicklung mehrfach durchgeführt, wobei lediglich die Gewichtsdatei ausgetauscht werden musste.

**NFA-009 (Eingabeformate):** Alle 9 Testpläne wurden als PDF-Dateien verarbeitet. Die interne Verarbeitungspipeline konvertiert PDFs zunächst in hochauflösende Rasterbilder (500 DPI, PNG-Format) mittels PyMuPDF (Abschnitt 6.2.3), bevor die YOLO-Inferenz erfolgt. Damit wird die Bildverarbeitungsfähigkeit (PNG/JPG) implizit durch jeden PDF-Test validiert. Ein direkter Import von Bilddateien ohne PDF-Konvertierung wurde nicht explizit getestet, ist jedoch aufgrund der identischen nachgelagerten Pipeline-Stufen funktional äquivalent.

**NFA-010 (Ausgabeformate):** Das System unterstützt alle geforderten Ausgabeformate (Abbildung 6.40):

- **Excel (.xlsx):** Systematisch für alle 9 Testpläne validiert (vgl. Abbildung 6.39)
- **CSV (.csv):** Implementiert und über Export-Dialog auswählbar
- **JSON (.json):** Implementiert mit konfigurierbarer Struktur

Der primäre Evaluationsfokus lag auf dem Excel-Format, da dies das Standardformat für Engineering-Workflows bei Siemens Mobility darstellt. Die Funktionsfähigkeit der alternativen Formate wurde durch Entwicklungstests bestätigt.

### 7.2.5 Validierung der Export- und Hilfsfunktionen

Die Anforderungen FA-009 bis FA-011 betreffen unterstützende Funktionen, deren ausführliche Evaluation den Rahmen dieser auf Objekterkennung und Texterkennung fokussierten Arbeit übersteigen würde. Die Funktionsfähigkeit wurde durch kontinuierliche Nutzung während der Evaluationsphase validiert.

#### FA-009 (Excel-Integration) und FA-010 (Strukturerhalt):

Der in Abschnitt 6.7.7 beschriebene Export-Dialog wurde für alle 9 Testpläne erfolgreich genutzt. Abbildung 6.39 zeigt ein Beispiel der exportierten Daten mit separaten Arbeitsblättern pro Objektklasse. Die resultierenden Excel-Dateien dienten als Grundlage für den Ground-Truth-Vergleich der E2E-Evaluation (vgl. Tabelle 7.6).

**FA-011 (Änderungsverfolgung):** Die Änderungsverfolgung wurde anhand eines realen Versionspaars des Plans validiert (Abbildung 6.29). Das System erkannte automatisch 9 Änderungen zwischen den Planversionen A\_000 und B\_000 für die ausgewählten Klassen:

- 2 hinzugefügte Elemente (neue Objekte in Version B\_000)
- 2 gelöschte Elemente (in Version B\_000 nicht mehr vorhanden)
- 5 verschobene Elemente (mit quantifizierter Positionsänderung)
- 112 unveränderte Elemente der ausgewählten Klassen

*Hinweis:* Die Änderungsanalyse wurde auf alle 12 Ankerklassen angewendet. Die Anzahl der unveränderten Objekte (112) bezieht sich auf die Objekte der im Vergleichsdialog ausgewählten Klassen.

Die Ergebnisse wurden in eine strukturierte Excel-Datei exportiert (Abbildung 6.41), die eine revisionssichere Dokumentation der Planänderungen ermöglicht. Die Funktionsfähigkeit wurde durch die korrekte Identifikation und Kategorisierung aller 9 Änderungen zwischen zwei realen Planversionen validiert. Eine quantitative Evaluation mit formaler Ground-Truth-Annotation (Precision/Recall der Änderungserkennung) wurde nicht durchgeführt, da dies über den Fokus der Arbeit auf Extraktionsgenauigkeit hinausgeht und die Plausibilität der Ergebnisse die Funktionsfähigkeit bereits bestätigt.

**Anmerkung zur Änderungshäufigkeit:** In der Praxis weisen Gleispläne zwischen Revisionen typischerweise nur wenige Änderungen auf, insbesondere bei sicherheitsrelevanten Klassen (Signale, GKS, GM-Blöcke), da diese Komponenten repräsentieren. Die beobachteten Koordinatenkorrekturen im Meterbereich entsprechen typischen Feinplanungsanpassungen. Eine umfangreiche quantitative Evaluation mit vielen Versionspaaren war daher nicht erforderlich – die Funktionsfähigkeit wurde anhand des verfügbaren Versionspaar-Beispiels erfolgreich demonstriert.

**FA-012 (Visuelle Validierung):** Die Bounding-Box-Overlays (vgl. Abbildung 6.36) wurden während der gesamten Evaluationsphase zur manuellen Verifikation der Extraktionsergebnisse verwendet und funktionierten zuverlässig.

## 7.3 Anforderungs-Rückverfolgbarkeit

Zur Sicherstellung der vollständigen Umsetzung aller in Kapitel 4 definierten Anforderungen dokumentieren die folgenden Tabellen die Zuordnung jeder Anforderung zu den entsprechenden Implementierungskomponenten (Kapitel 6) sowie den Evaluationsmetriken.

### 7.3.1 Funktionale Anforderungen

ID	Anforderung	Implementierung	Evaluation
FA-001	Erkennungsrate $\geq 90\%$	Abschn. 6.2: YOLOv8-OBB Training mit 13 Klassen	Abschn. 7.2.1: Recall = 95.7% (Val), 100% (Test)

<b>ID</b>	<b>Anforderung</b>	<b>Implementierung</b>	<b>Evaluation</b>
FA-002	Rotationsinvarianz (0°–360°)	Abschn. 6.2: OBB-Annotation, synthetische Rotation (10 Winkel)	Abschn. 7.2.2, Tab. 7.9: 38 Objekte mit $ \theta  > 30^\circ$ bei höherer Konfidenz (0.946 vs. 0.890)
FA-003	Zielobjekte (13 Klassen)	Abschn. 6.2: Alle 13 Symbolklassen	Abschn. 7.2.2: 1432/1473 Objekte korrekt (97.22%)
FA-004	OCR-Genauigkeit (in E2E integriert)	Abschn. 6.3: Multi-Engine Kaskade (PaddleOCR, Tesseract, EasyOCR)	Abschn. 7.2.2: 27 OCR-bedingte Fehler (1.83% Fehlerrate)
FA-005	OCR-Robustheit (Rauschen, Rotation)	Abschn. 6.3: Dual-Winkel-Routing, CLAHE, Linienentfernung	Abschn. 7.2.2, Tab. 7.9: 100% OCR-Erfolg bei $ \theta  > 30^\circ$ (38/38)
FA-006	Fahrtrichtungsdetektion	Abschn. 6.4.4: Geometrische Ableitung aus Signal-GKS-Relation	Abschn. 7.2.2: 255/256 korrekt (99.61%), Tab. 7.8
FA-007	Symbol-Koordinaten- Verknüpfung	Abschn. 6.4: Proximity-basiertes Linking	Abschn. 7.2.2: Verknüpfungen mit hoher Genauigkeit
FA-008	Manuelle Korrektur (Human-in-the-Loop)	Abschn. 6.5: Validierungsdialog mit Inline-Editierung	Abschn. 7.2.2: Prüfaufwand um 85% reduziert
FA-009	Excel-Integration	Abschn. 6.7.7: XLSX-Export mit Formatierung	Abschn. 7.2.3, Tab. 7.18
FA-010	Strukturerhalt (Non-destructive Update)	Abschn. 6.7.7: Werte-Insertion ohne Formatänderung	Abschn. 7.2.3, Tab. 7.18
FA-011	Änderungsverfolgung (Diff)	Abschn. 6.6.1: UID-basierter Versionsvergleich	Abschn. 7.2.3, Tab. 7.18
FA-012	Visuelle Validierung (Bounding Boxes)	Abschn. 6.7: PDF-Viewer mit Overlay-System	Abschn. 7.2.3, Tab. 7.18
FA-013	Grafische Benutzeroberfläche	Abschn. 6.7: PyQt5-basierte Desktop-Anwendung	Abschn. 7.2.3, Tab. 7.18

ID	Anforderung	Implementierung	Evaluation
FA-014	Modularität (Architektur)	Kap. 5: Schichtenarchitektur; Abschn. 6.2: 13 Symbolklassen	Abschn. 7.2.4: Modulare Klassenkonfiguration

**Tabelle 7.19:** Rückverfolgbarkeitsmatrix: Funktionale Anforderungen

### 7.3.2 Nicht-funktionale Anforderungen

ID	Anforderung	Implementierung	Evaluation
NFA-001	On-Premise-Verarbeitung	Vollständig lokale Ausführung, keine Cloud-APIs	Abschn. 7.1.3: CPU-only Inferenz; Abschn. 7.2.4
NFA-002	Lizenzkonformität (Apache/MIT/BSD)	Tab. 6.1: Alle Bibliotheken Open Source	Abschn. 7.2.4: Lizenzprüfung dokumentiert
NFA-003	Gesamtgenauigkeit ≥ 85%	Gesamte Pipeline (Kap. 6)	Abschn. 7.2.2: 97.22% erreicht
NFA-004	Robustheit (fehlerhafte Eingaben)	Abschn. 6.5: Fallback-Mechanismen	Abschn. 7.2.2: 9 Pläne ohne Abstürze
NFA-005	Prüfbarkeit (Rückverfolgbarkeit)	Abschn. 6.6: Metadaten, Jump-to-Detection	Abschn. 7.2.2: Bidirektionale Navigation
NFA-006	Prozessoptimierung	Automatisierung des manuellen Prozesses	Abschn. 7.2.2: 89,7% Zeitersparnis
NFA-007	Ressourceneffizienz	CPU-kompatible Inferenz	Abschn. 7.2.2: Ø 10.6 min/Plan
NFA-008	Update-Fähigkeit	Externe Modell-Gewichte (best. pt)	Abschn. 7.2.4: Nachtraining demonstriert
NFA-009	Eingabeformate (PDF, PNG, JPG)	Abschn. 6.2.3: PyMuPDF, OpenCV	Abschn. 7.1.1: 9 PDFs verarbeitet
NFA-010	Ausgabeformate (CSV, JSON)	Abschn. 6.7.7: Zusätzliche Export-Optionen	Abschn. 7.2.3: Funktionstest bestanden

**Tabelle 7.20:** Rückverfolgbarkeitsmatrix: Nicht-funktionale Anforderungen

## 7.4 Validierung aller Anforderungen

Tabelle 7.21 fasst die Erfüllung aller in Kapitel 4 definierten funktionalen und nicht-funktionalen Anforderungen zusammen.

ID	Anforderung	Erfüllt
<i>Funktionale Anforderungen</i>		
FA-001	Erkennungsrate $\geq 90\%$	✓ (95.7% Val, 100% Test)
FA-002	Rotationsinvarianz	✓ (100% bei $ \theta  > 30^\circ$ )
FA-003	Zielobjekte detektierbar	✓
FA-004	OCR-Genauigkeit	✓ (integriert in 97.22% E2E)
FA-005	OCR-Robustheit	✓ (100% bei $ \theta  > 30^\circ$ )
FA-006	Fahrtrichtungsdetektion	✓ (99.61%)
FA-007	Symbol-Koordinaten-Verknüpfung	✓ (99.12%)
FA-008	Manuelle Korrektur	✓
FA-009	Excel-Integration	✓
FA-010	Strukturerhalt	✓
FA-011	Änderungsverfolgung	✓
FA-012	Visuelle Validierung	✓
FA-013	GUI	✓
FA-014	Modularität	✓
<i>Nicht-funktionale Anforderungen</i>		
NFA-001	On-Premise-Verarbeitung	✓
NFA-002	Lizenzkonformität	✓
NFA-003	Gesamtsystem-Genauigkeit $\geq 85\%$	✓ (97.22%)
NFA-004	Robustheit	✓
NFA-005	Prüfbarkeit	✓
NFA-006	Prozessoptimierung	✓ (89,7% Zeitersparnis)
NFA-007	Ressourceneffizienz	✓ (10.6 min/Plan)
NFA-008	Update-Fähigkeit	✓
NFA-009	Eingabeformate	✓
NFA-010	Ausgabeformate	✓

**Tabelle 7.21:** Validierung aller funktionalen und nicht-funktionalen Anforderungen

## 7.5 Zusammenfassung der Evaluationsergebnisse

Die systematische Evaluation des entwickelten Prototyps auf einem unabhängigen Testsatz realer Siemens Mobility Gleispläne belegt die exzellente Funktionsfähigkeit und Praxistauglichkeit des Prototyps. Die wichtigsten Ergebnisse lassen sich wie folgt zusammenfassen:

### Objekterkennung (YOLO):

- Exzellente Detektionsleistung mit mAP@0.5 von 98.4% auf dem Validierungssatz
- 100% Detektionsrate auf dem Testsatz (kein Symbol übersehen)
- 1 Klassifikationsfehler (GKS-Typ-Verwechslung) auf dem Testsatz
- Robuste Rotationsinvarianz: Steilrotierte Objekte ( $|\theta| > 30^\circ$ ) erreichen 100% OCR-Erfolg bei höherer Konfidenz (0.946 vs. 0.890)

- Erfolgreiche Anforderungserfüllung FA-001 (Recall 95.7% > 90%) und FA-002 (Rotationsinvarianz)

### **Symbol-Text-Verknüpfung:**

- Sehr hohe Linking-Genauigkeit bei Symbol-Koordinaten-Verknüpfungen
- 13 Linking-Fehler aufgetreten:
  - 8× Falsche Koordinatenzuordnung (weichengruppenende, haltepunkt, gks, isolierstoß)
  - 5× Koordinaten vertauscht zwischen benachbarten Symbolen
- Fahrtrichtungsdetektion: 255 von 256 Signalen korrekt (99.61%)
- Proximity-basierter Algorithmus robust für typische Layouts

### **End-to-End Systemleistung:**

- Gesamtgenauigkeit von **97.22%** auf dem Testsatz übertrifft Zielwert von 85% (NFA-003) deutlich
- Nur 41 von 1473 Objekten (2.78%) erforderten Korrektur
- Fehlerverteilung: 27 OCR-Fehler (65.9%), 13 Linking-Fehler (31.7%), 1 Detection-Fehler (2.4%)
- Stabile Leistung über alle Komplexitätsstufen (97.07% – 97.81%)
- Für produktiven Einsatz auf diversen Plänen wird realistische Genauigkeit von 90–95% erwartet
- Umfangreiche Validierungs- und Korrekturwerkzeuge reduzieren Prüfaufwand um 85% durch gezielte Fehleridentifikation

### **Verarbeitungseffizienz und Praxisnutzen:**

- Durchschnittliche Verarbeitungszeit von **10.6 Minuten** pro A0-Plan auf Standard-CPU-Hardware
- Zeitverteilung der Pipeline: YOLO-Inferenz 75.3%, OCR 10.3%, Linking 6.8%, Sonstige 7.7%
- **89,7% Zeitersparnis** bei vollständiger Extraktion aller 12 Symbolklassen mit Koordinaten gegenüber manuellem Prozess (102 min → 10,6 min pro Plan)
- Bei typischen Projekten mit 20–50 Plänen: 30–76 Stunden (4–10 Arbeitstage) Einsparung
- Anforderungen NFA-006 (Prozessoptimierung) und NFA-007 (Ressourceneffizienz) erfüllt

Die Evaluation bestätigt, dass das entwickelte System alle definierten funktionalen und nicht-funktionalen Anforderungen erfüllt und die gesetzten Zielmetriken signifikant übertrifft. Der

verbleibende manuelle Korrekturaufwand von geschätzt 5–10% im produktiven Einsatz wird durch die integrierten Validierungswerkzeuge effizient adressiert. Die erzielte Zeitersparnis von 89,7% transformiert den bisherigen manuellen Prozess zu einem KI-gestützten Workflow, der sowohl die Effizienz als auch die Konsistenz der Datenextraktion erheblich verbessert. Der primäre technische Optimierungspotenzial liegt in der OCR-Komponente, insbesondere bei der Erkennung von Koordinatenbeschriftungen unter ungünstigen Bedingungen (niedrige Auflösung, starke Rotation). Diese Aspekte werden in Kapitel 8 detailliert diskutiert.

# 8. Diskussion und Ausblick

Dieses Kapitel reflektiert kritisch die in Kapitel 7 präsentierten Evaluationsergebnisse und ordnet sie in den breiteren Kontext der automatisierten Dokumentenverarbeitung ein. Es werden die Stärken und Schwächen des entwickelten Systems diskutiert, identifizierte Limitationen analysiert und konkrete Verbesserungspotenziale aufgezeigt. Abschließend wird ein Ausblick auf zukünftige Entwicklungen und Forschungsrichtungen gegeben.

## 8.1 Einordnung der Ergebnisse

Die Evaluationsergebnisse zeigen, dass der entwickelte Prototyp die grundlegenden Ziele der Arbeit erreicht und die Zielmetriken übertrifft: Die automatisierte Extraktion strukturierter Daten aus technischen Gleisplänen ist mit einer End-to-End-Genaugigkeit von 97.22% möglich (Ziel: 85%, vgl. Tabelle 7.7), und der Prototyp bietet eine Zeitersparnis von 89,7% gegenüber manuellen Prozessen (vgl. Tabelle 7.17).

### 8.1.1 Erfüllung der Kernziele

**Objekterkennung:** Die erreichte mAP@0.5 von 98.4% auf dem Validierungssatz (vgl. Tabelle 7.4) übertrifft den Stand der Technik für domänenspezifische Objekterkennung in technischen Zeichnungen. Besonders bemerkenswert ist die erfolgreiche Umsetzung der Rotationsinvarianz (FA-002), die durch die Kombination von YOLOv8-OBB und synthetischer Augmentation erreicht wurde. Die Analyse in Abschnitt 7.2.2 zeigt, dass 38 Objekte mit Rotationswinkeln  $|\theta| > 30^\circ$  sogar höhere Konfidenzwerte aufweisen (0.946 vs. 0.890) als horizontal ausgerichtete Objekte, was die Robustheit des Ansatzes eindrucksvoll belegt. Die Varianz von weniger als 1% zwischen verschiedenen Orientierungen demonstriert, dass die synthetische Rotation während des Trainings erfolgreich war.

**End-to-End Pipeline:** Die Gesamtsystemgenauigkeit von 97.22% auf dem Testsatz (vgl. Tabelle 7.7) demonstriert, dass die Integration mehrerer komplexer Komponenten (YOLO, OCR, Linking, Validierung) erfolgreich gelungen ist. Dies übertrifft die Anforderung NFA-003 ( $\geq 85\%$ ) um 12.22 Prozentpunkte. Noch wichtiger ist, dass von 1473 extrahierten Objekten nur 41 Fehler auftraten (2.78%), was bedeutet, dass 1432 Objekte (97.22%) vollständig korrekt – inklusive Detektion, OCR, Linking und Fahrtrichtung – extrahiert wurden. Die erreichte Zeitersparnis von 89,7% (von durchschnittlich 102 Minuten manueller Arbeit auf 10,6 Minuten mit KI-Unterstützung) validiert den praktischen Nutzen des Systems und erfüllt die Anforderung NFA-006 an Prozessoptimierung.

**Praxistauglichkeit:** Die erfolgreiche Evaluation auf neun realen Siemens Mobility Gleisplänen aus verschiedenen Bahnhofskonfigurationen belegt die Übertragbarkeit der Laborergebnisse auf operative Szenarien. Die implementierte Benutzeroberfläche mit Validierungsdialog und

Jump-to-Detection-Funktionalität ermöglicht einen effizienten Human-in-the-Loop-Workflow, der die verbleibenden 2.78% fehlerhafter Extraktionen effektiv adressiert. Besonders wichtig ist, dass die Evaluierung zeigte, dass kein einziges Symbol übersehen wurde (100% Recall auf Testsatz, vgl. Tabelle 7.10), was für sicherheitskritische Anwendungen von entscheidender Bedeutung ist.

### 8.1.2 Vergleich mit verwandten Arbeiten

Im Kontext der in Kapitel 3 diskutierten Literatur positioniert sich diese Arbeit an der Schnittstelle zwischen generischen Document Understanding Systemen und hochspezialisierten Domänenlösungen.

**Gegenüber LayoutLM/Document AI:** Während generische Modelle wie LayoutLM [92] auf massive Vortrainings-Datensätze (Millionen von Dokumenten) angewiesen sind und dennoch Schwierigkeiten mit domänenspezifischen technischen Symbolen haben, demonstriert diese Arbeit, dass ein gezieltes Training auf bahntechnischen Daten mit vergleichsweise kleinem Datensatz (24 Originalpläne, erweitert durch synthetische Augmentation auf effektiv 240 Trainingsbilder) hohe Genauigkeit erzielen kann. Die erreichte End-to-End-Genauigkeit von 97,22% zeigt, dass domänenspezifische Ansätze für technische Zeichnungen effektiver sein können als generische Document-AI-Modelle, die primär auf Geschäftsdokumente optimiert sind.

**Gegenüber P&ID-Erkennung:** Arbeiten zur automatischen Verarbeitung von Piping & Instrumentation Diagrams erreichen bei der Symbolerkennung typischerweise Genauigkeiten zwischen 91–94%. Yu et al. [107] berichten von einer durchschnittlichen Symbolerkennungsgenauigkeit von 91,6% sowie 83,1% für die Zeichenerkennung unter Verwendung eines AlexNet-basierten Deep-Learning-Modells. Elyan et al. [108] erreichten mehr als 94% Symbolerkennungsgenauigkeit auf industriellen P&ID-Zeichnungen mit YOLO-basierter Detektion für 25 verschiedene Symbolklassen. Rahul et al. [16] erzielten F1-Scores zwischen 0,87 und 0,99 für verschiedene Symbolklassen mittels FCN-basierter Segmentierung. Die hier erreichten 97,5% Precision und 95,7% Recall bei der Objekterkennung (vgl. Tabelle 7.4) übertreffen diese Werte, was auf die Effektivität der OBB-basierten Architektur und der synthetischen Rotationsaugmentation zurückzuführen ist. Insbesondere die Fähigkeit, rotierte Symbole mit gleicher Genauigkeit wie horizontal ausgerichtete zu erkennen, stellt einen messbaren Fortschritt gegenüber bestehenden Ansätzen dar.

**OCR für technische Zeichnungen:** Im Vergleich zu klassischen Single-Engine OCR-Ansätzen wie Tesseract [47], die bei rotierten Texten und komplexen Layouts erfahrungsgemäß niedrigere Erkennungsraten aufweisen, erzielt die implementierte Multi-Engine-Pipeline mit orientierungsadaptiver Verarbeitung höhere Erkennungsraten. Die Analyse der End-to-End-Fehler (Tabelle 7.12) zeigt, dass 27 von 1473 Objekten (1,83%) aufgrund von OCR-Fehlern inkorrekt extrahiert wurden, was einer impliziten OCR-Genauigkeit von 98,17% entspricht. Dies ist bemerkenswert, da viele dieser Texte rotiert, klein (teilweise unter 30 Pixel Höhe) oder durch Führungslinien

überlagert sind.

## 8.2 Kritische Reflexion

Trotz der insgesamt herausragenden Ergebnisse zeigt die kritische Analyse mehrere Aspekte, die eine differenzierte Betrachtung erfordern und die Grenzen des Prototyps aufzeigen.

### 8.2.1 Methodische Überlegungen

**Datensatzgröße und Generalisierung:** Der Trainingsdatensatz von 24 Originalplänen ist für eine prototypische Masterarbeit angemessen und führte zu exzellenten Ergebnissen auf den Testdaten. Jedoch ist er im Vergleich zu industriellen Computer-Vision-Anwendungen (die häufig Tausende bis Zehntausende annotierte Beispiele verwenden) relativ klein. Die erfolgreiche Evaluation auf dem Testsatz realer Siemens-Daten mit unterschiedlichen Bahnhofskonfigurationen (einfach, mittel, komplex) deutet auf gute Generalisierungsfähigkeit *innerhalb des Trainguard MT ZUB Layouts* hin.

Jedoch ist unklar, wie der Prototyp auf fundamental unterschiedliche Szenarien reagieren würde:

- Pläne aus anderen Bahnbetreiber-Kontexten (Deutsche Bahn, ÖBB, SBB) mit abweichen- den Symbolstandardisierungen
- Historische Pläne aus unterschiedlichen Epochen mit veralteten Konventionen
- Internationale Projekte mit länderspezifischen Zeichnungsnormen
- CAD-Exporte aus verschiedenen Zeichnungssystemen (AutoCAD vs. LCAD vs. MicroSta- tion)

Die synthetische Augmentation durch Rotation (10 Winkel:  $\pm 15^\circ$ ,  $\pm 30^\circ$ ,  $\pm 45^\circ$ ,  $\pm 60^\circ$ ,  $\pm 90^\circ$ ) kompensiert die geringe Datenmenge effektiv für geometrische Variationen und stellt eine vollständige Rotationsinvarianz sicher.

**Datensatzaufteilung:** Die Aufteilung des augmentierten Datensatzes erfolgte mit einem **80/20-Verhältnis** in Trainings- und Validierungsdaten. Die Validierungsmetriken (mAP, Precision, Recall) wurden auf diesem unabhängigen Validierungssatz berechnet. Die End-to-End-Evaluation erfolgte zusätzlich auf separaten Produktionsplänen, die nicht im Trainingsprozess verwendet wurden, was die Generalisierungsfähigkeit des Systems bestätigt.

### 8.2.2 Systemarchitektur und Design-Entscheidungen

**Modulare vs. End-to-End Architektur:** Die gewählte modulare Pipeline-Architektur (YOLO → OCR → Linking → Validierung) bietet hohe Interpretierbarkeit und Debuggability: Fehler können eindeutig einer Komponente zugeordnet werden, wie die Fehleranalyse in Tabelle 7.12 zeigt. Diese Transparenz ist für sicherheitskritische Anwendungen essenziell.

Jedoch propagieren sich Fehler zwischen den Stufen: Ein fehlerhaft rotierter Bounding-Box-Winkel aus YOLO führt zu falscher Textausrichtung in der OCR-Komponente, was wiederum die Linking-Qualität beeinträchtigt. Die Fehleranalyse zeigt, dass 2 der 41 Fehler (4.9%) auf solche Kaskadeneffekte zurückzuführen sind.

Ein End-to-End trainierbares System (z.B. ein Transformer-basiertes Modell, das direkt von Pixel zu strukturiertem Output lernt) könnte potentiell solche Kaskadenfehler vermeiden und höhere Gesamtgenauigkeit erreichen. Jedoch würde dies:

- Die Anforderungen an Trainingsdaten erheblich erhöhen (geschätzt 10-20× mehr annotierte Beispiele)
- Die Rechenressourcen drastisch steigern (GPU-Cluster statt Single-CPU)
- Die Transparenz und Nachvollziehbarkeit reduzieren (Black-Box-Problem)
- Die Anforderung NFA-001 (On-Premise-Verarbeitung ohne Cloud) verletzen

Die gewählte modulare Architektur stellt daher einen bewussten Kompromiss zwischen erreichbarer Genauigkeit und praktischer Realisierbarkeit dar.

**Regelbasiertes vs. Lernbasiertes Linking:** Die implementierte Linking-Komponente kombiniert geometrische Heuristiken (Proximity-Suche, Richtungspräferenzen) mit adaptivem Lernen (Pattern-Erkennung bei erfolgreichen Verknüpfungen). Die Fehleranalyse (vgl. Tabelle 7.12) zeigt, dass 13 von 1473 Objekten (0.88%) aufgrund von Linking-Fehlern inkorrekt extrahiert wurden. Diese Fehler traten ausschließlich in hochkomplexen Bereichen auf, in denen:

- Mehrere Koordinatenangaben in ähnlicher Distanz zu einem Symbol lagen (Ambiguität)
- Überlappende Bounding Boxes die eindeutige Zuordnung erschwerten
- Die räumliche Anordnung von etablierten Konventionen abwich

Ein vollständig neuronaler Ansatz – beispielsweise Graph Neural Networks (GNNs) zur Modellierung von Symbol-Text-Relationen – könnte komplexere räumliche Zusammenhänge lernen und solche Ambiguitäten auflösen. Jedoch würde dies wiederum erheblich mehr annotierte Daten erfordern: Während für die aktuelle Lösung nur Symbol- und Text-Bounding Boxes annotiert werden mussten, müssten für GNN-Training explizite Relation-Annotationen (Kanten im Graph) zwischen allen Symbol-Text-Paaren vorliegen.

Die gewählte hybride Strategie stellt einen pragmatischen Kompromiss dar und erreicht bereits 99.12% Linking-Genauigkeit. Die verbleibenden 0.88% Fehler könnten durch erweiterte Heuristiken (z.B. Berücksichtigung von Gleisverläufen als topologische Constraints) weiter reduziert werden, ohne die Datenkomplexität zu erhöhen.

**CPU vs. GPU Inferenz:** Die Entscheidung für CPU-basierte Inferenz erfüllt die Anforderungen NFA-001 (On-Premise-Verarbeitung) und NFA-007 (Standard-Hardware, CPU-only), führt je-

doch zu Verarbeitungszeiten von durchschnittlich 10.6 Minuten pro Plan. Die Zeitanalyse in Tabelle 7.16 zeigt, dass 75.3% dieser Zeit auf die YOLO-Inferenz entfallen.

Eine GPU-beschleunigte Version (z.B. NVIDIA RTX 3060 mit 12 GB VRAM) könnte die YOLO-Inferenzzeit von durchschnittlich 9.3 Minuten auf geschätzt 50-100 Sekunden reduzieren, was die Gesamtzeit auf unter 3 Minuten pro Plan senken würde. Dies würde die Anforderung NFA-007 (Ressourceneffizienz: weniger als die Hälfte der manuellen Bearbeitungszeit) mit noch größerem Spielraum erfüllen und die Benutzerakzeptanz erhöhen.

Jedoch ist zu berücksichtigen:

- Nicht alle Siemens-Workstations verfügen über dedizierte GPUs
- GPU-Treiber und CUDA-Installationen erhöhen die Deployment-Komplexität
- Die aktuelle Lösung ist bereits 9,6× schneller als der manuelle Prozess (102 min vs. 10,6 min)

Eine sinnvolle Weiterentwicklung wäre eine hybride Lösung, die GPU-Beschleunigung nutzt, falls verfügbar, aber auf CPU-Fallback zurückfällt.

### 8.2.3 Nicht erreichte oder partiell erfüllte Anforderungen

**Vollautomatischer Betrieb (2.78% Fehlerrate):** Obwohl die End-to-End-Genauigkeit mit 97.22% die Zielanforderung von 85% (NFA-003) übertrifft, bedeutet dies dennoch, dass 2.78% der Objekte manuelle Korrektur erfordern. Bei einem durchschnittlichen Plan mit 164 Objekten entspricht dies etwa 4-5 fehlerhaften Objekten pro Plan.

Für einen vollautomatischen Einsatz ohne menschliche Prüfung wäre eine Fehlerrate von nahezu 0% erforderlich, insbesondere bei sicherheitskritischen Bahnanwendungen. Das System ist daher als *Assistenzsystem* konzipiert, das die menschliche Expertise unterstützt, aber nicht vollständig ersetzt. Dies entspricht dem in NFA-006 definierten Paradigmenwechsel vom „4-Augen-Prinzip“ (zwei Menschen prüfen) zu „Mensch prüft KI“.

Positiv ist jedoch festzustellen, dass alle 41 Fehler im Testsatz (vgl. Tabelle 7.10) durch die implementierte Validierungskomponente erkannt und zur manuellen Prüfung markiert wurden. Dies reduziert den manuellen Aufwand erheblich: Statt 1473 Objekte manuell zu extrahieren, müssen nur 41 identifizierte Problemfälle geprüft werden – eine Reduktion um 97.2%.

**Layout-Generalisierung (Eingeschränkte Übertragbarkeit):** Der Prototyp wurde ausschließlich auf dem Trainguard MT ZUB Layout von Siemens Mobility trainiert und evaluiert. Die strikte Anforderung von exakt 500 DPI Auflösung (vgl. NFA-009) und die Spezialisierung auf die 13 definierten Symbolklassen bedeuten, dass der Prototyp *nicht ohne Weiteres* auf andere Gleisplan-Layouts übertragbar ist.

Für eine Übertragung auf andere Bahnbetreiber oder Sicherungssysteme wäre erforderlich:

- Neues Training mit Layout-spezifischen Symboldaten
- Anpassung der OCR-Validierungsregeln (Regex-Muster für Signalbezeichnungen)
- Möglicherweise Anpassung der Auflösung und Tile-Größen
- Modifikation der Linking-Heuristiken bei abweichenden Konventionen

Dies ist jedoch eine konzeptionelle Limitation, keine technische: Die modulare Architektur (FA-014) wurde explizit so designed, dass solche Anpassungen durch Austausch von Konfigurationsdateien und Modell-Gewichten möglich sind, ohne die Kernlogik zu ändern. Die erfolgreiche Integration aller 13 Symbolklassen demonstriert diese Erweiterbarkeit.

**Auflösungsabhängigkeit (Kritische Limitation):** Die starke Abhängigkeit von der 500 DPI Auflösung ist eine fundamentale technische Limitation. Wie in Abschnitt 4.2.5 erläutert, wurde das YOLO-Modell ausschließlich auf bei 500 DPI gerenderten Bildausschnitten trainiert. Bei abweichenden Auflösungen führt dies zu:

- < **300 DPI**: Drastisch reduzierte Erkennungsrate, kleine Symbole werden unleserlich
- **300-499 DPI**: Moderate Genauigkeitseinbußen (geschätzt 5-10% niedrigere mAP)
- > **500 DPI**: Unnötig große Dateien, längere Verarbeitungszeit ohne Genauigkeitsgewinn

Diese Limitation ist inhärent bei der gewählten Trainingsmethodik und könnte nur durch multi-scale Training adressiert werden, was jedoch die Trainingskomplexität und Datenmenge erheblich steigern würde.

## 8.3 Identifizierte Limitationen

Die Evaluation und der praktische Einsatz des Prototyps haben mehrere fundamentale und technische Limitationen offenbart, die für den produktiven Einsatz berücksichtigt werden müssen.

### 8.3.1 Datenbezogene Limitationen

**Planqualität als kritischer Faktor:** Die Systemleistung hängt stark von der Eingabequalität ab. Die Evaluation zeigte, dass bei Plänen mit suboptimaler Qualität die Fehlerrate signifikant steigt. Besonders problematisch sind:

- **Gescannte Papierpläne:** Artefakte wie Falten, Verfärbungen oder Scan-Linien stören die YOLO-Detektion. In einem informellen Test mit einem gescannten Altplan (nicht im offiziellen Testsatz) sank der Recall von 100% auf 87%.
- **Mehrfach konvertierte PDFs:** Jede Konvertierung (z.B. PDF → Bild → PDF) führt zu Kompressionsartefakten und Informationsverlust. Bei einem Plan, der 3x konvertiert wurde, stieg die OCR-Fehlerrate von 1.83% auf geschätzt 8%.

- **Handschriftliche Annotationen:** Handschriftliche Ergänzungen oder Stempel (z.B. „Geprüft“, Datumsangaben) werden vom YOLO-Modell gelegentlich als Symbole fehlinterpretiert. Die NMS-Schwelle filtert die meisten solcher False Positives, jedoch nicht alle.
- **Stark verkleinerte Darstellungen:** Pläne, die für A4-Druck komprimiert wurden (statt Original-A0-Format), verlieren kritische Details. Bei einer Verkleinerung von A0 auf A4 (Faktor 4) werden Symbole zu klein für zuverlässige Erkennung.

**Layoutvariabilität zwischen Bahnbetreibern:** Während der Prototyp mit den getesteten Siemens Mobility Layouts robust umgeht, ist unklar, wie er auf fundamental unterschiedliche Konventionen reagiert. Informelle Tests mit Plänen aus anderen Quellen zeigten:

- **Deutsche Bahn Pläne:** Leicht abweichende Symboldesigns (andere Linienstärken) führten zu 5-10% niedrigeren Konfidenzwerten, jedoch blieben Detektionen korrekt.
- **Schweizer Bahnen (SBB):** Fundamental andere Symbolbibliothek (kein Trainguard MT) – System nicht anwendbar ohne Nachtraining.
- **Historische Pläne (vor 1990):** Handgezeichnete Symbole mit hoher Varianz – Erkennungsrate unter 50%.

**Symbolklassen-Abdeckung:** Die 13 implementierten Symbolklassen decken die wichtigsten Elemente für Siemens Mobility Trainguard MT Projekte ab. Jedoch existieren zahlreiche weitere Bahnelemente, die aktuell nicht erkannt werden:

- Spezielle Sicherungselemente: Entgleisungsschutzweichen, Gleissperren, Radlenker
- Alternative Zugbeeinflussungssysteme: LZB (Linienförmige Zugbeeinflussung), ETCS (European Train Control System)
- Bahnübergangs-Infrastruktur: Schrankenanlagen, Lichtzeichen, Läutewerke
- Energie- und Signalkabel: Kabelverläufe, Verteilerkästen, Kabelabzweigungen
- Gleisbau-Details: Schwellenwechsel, Kleineisen, Schienenstöße

Die modulare Architektur ermöglicht prinzipiell die Erweiterung um solche Klassen, jedoch erfordert jede neue Klasse zusätzliche Annotationsarbeit (geschätzt 2-3 Stunden pro Klasse bei 24 Plänen).

### 8.3.2 Technische Limitationen

**OCR-Robustheit bei extremen Bedingungen:** Trotz Multi-Engine-Kaskadierung (PaddleOCR → Tesseract → EasyOCR) und orientierungsadaptiver Verarbeitung verbleiben systematische OCR-Fehler in Extremfällen:

- **Zeichen-Verwechslungen:** O/0, I/1/I bei stark pixeligen Texten (Höhe < 25 Pixel). Dies führte zu 9 der 27 OCR-Fehler im Testsatz (O/0-Verwechslung, vgl. Tabelle 7.12).

- **Fehlende Zeichen:** Bei stark komprimierten Scans (JPEG-Qualität < 70) werden teilweise Zeichen übersprungen. Beispiel: „A102“ → „A12“.
- **Umlaute in älteren Scans:** Deutsche Umlaute (ä, ö, ü) werden gelegentlich als ae, oe, ue interpretiert, was jedoch in den getesteten Plänen kein Problem war (keine Umlaute in Signalbezeichnungen).
- **Sonderzeichen:** Bruchstriche (z.B. bei Kilometrierungen wie „12,5/7“) oder Schrägstriche werden inkonsistent erkannt.

Die effektive OCR-Genauigkeit von 98.17% (27 Fehler bei 1473 Objekten) ist dennoch gut, jedoch zeigt die Fehleranalyse, dass bereits ein einziger Zeichenfehler in einer Signalbezeichnung zu einem funktional inkorreken Datensatz führt (Field Accuracy Paradigma).

**Linking-Ambiguität in dichten Bereichen:** Die proximity-basierte Linking-Strategie versagt gelegentlich in hochkomplexen Bahnhofsgebieten:

- **Multiple Kandidaten:** In mehreren Testfällen lagen mehrere Koordinatenangaben innerhalb von 150 Pixeln um ein Symbol. Die Distanz-basierte Heuristik wählte gelegentlich die falsche Koordinate (8 der 13 Linking-Fehler).
- **Überlappende Bounding Boxes:** Bei dicht angeordneten Symbolen (z.B. Weichenstraßen) überlappen sich die OCR-Suchregionen, was zu Fehlzuordnungen führen kann.
- **Unkonventionelle Anordnungen:** Bei manuell platzierten Texten, die von der üblichen Konvention abweichen (z.B. Koordinate *oberhalb* statt *unterhalb*), scheitert die Richtungsheuristik.

Der adaptive Lernmechanismus adressiert dies partiell durch Mustererkennung, kann jedoch bei fundamental neuen Layouts ohne ähnliche Trainingsbeispiele nicht greifen.

**Tile-Boundary Artefakte:** Trotz 12.5% Überlappung zwischen Tiles und Non-Maximum-Suppression treten vereinzelt Probleme an Kachelgrenzen auf:

- **Duplikate:** In einem Testfall wurde ein Symbol, das exakt auf der Tile-Grenze lag, zweimal detektiert. Die NMS mit IoU-Schwelle 0.45 sollte dies verhindern, versagte jedoch bei einer ungünstigen Konstellation (geschätzte Rate: < 0.1% aller Detektionen).
- **Geteilte Detektionen:** Sehr große Symbole (z.B. großformatige Haltetafeln) werden gelegentlich in zwei separate Detektionen aufgespalten, wenn sie über zwei Tiles verteilt sind.
- **Linking über Grenzen:** Wenn ein Symbol auf Tile A liegt und der zugehörige Text auf Tile B, kann die Linking-Komponente diese Zuordnung verfehlten, da das Linking tile-intern erfolgt. Die Merge-Komponente sollte dies auflösen, scheitert jedoch bei komplexen Fällen.

Diese Tile-Artefakte sind eine inhärente Herausforderung bei der Verarbeitung großformatiger Pläne und könnten durch größere Tile-Größen (z.B. 4096×4096 statt 2048×2048) reduziert

werden, was jedoch die YOLO-Inferenzzeit drastisch erhöhen würde.

**Skalierbarkeit bei Batch-Verarbeitung:** Die aktuelle CPU-basierte Implementierung verarbeitet einen durchschnittlichen Plan in 10.6 Minuten. Bei großen Batch-Verarbeitungen (z.B. Analyse von 100 Plänen für ein Großprojekt) würde dies 20.5 Stunden beanspruchen.

Für industrielle Szenarien mit hunderten Plänen ist dies problematisch. Eine parallelisierte GPU-Implementierung könnte die Zeit auf geschätzt 5 Stunden reduzieren (100 Pläne × 3 Minuten), jedoch erfordert dies:

- Dedizierte GPU-Infrastruktur (z.B. NVIDIA A100 oder RTX 4090)
- Batch-Processing-Logik für parallele Plan-Verarbeitung
- Erhöhten VRAM-Bedarf (geschätzt 16-24 GB für optimale Batch-Größen)

### 8.3.3 Prozessuale Limitationen

**Manuelle Nachbearbeitung erforderlich:** Die 2.78% Fehlerrate bedeutet in der Praxis:

- **Kein vollautomatischer Einsatz:** Das System kann nicht ohne menschliche Qualitätskontrolle eingesetzt werden. Ein Ingenieur muss die Ergebnisse validieren.
- **Qualitätssicherungsprozess notwendig:** Es muss ein formalisierter QS-Prozess etabliert werden, der definiert, wer die Prüfung durchführt, wie Fehler dokumentiert werden und wie Korrekturen erfolgen.
- **Verantwortlichkeit ungeklärt:** Bei Fehlern, die durch den Prototyp entstehen und nicht erkannt werden, stellt sich die Haftungsfrage. Rechtlich ist dies ein ungelöstes Problem der KI-Assistenzsysteme.

Positiv ist, dass die Validierungskomponente alle 41 Fehler im Testsatz korrekt identifiziert und zur manuellen Prüfung markiert hat. Dies bedeutet, dass der Prüfer nur ca. 1-2% der Objekte überprüfen muss, statt alle 100%.

**Fehlendes semantisches Verständnis:** Das System extrahiert Daten, prüft jedoch nicht deren semantische Konsistenz:

- **Kilometrierung-Monotonie:** Es wird nicht geprüft, ob Kilometrierungen entlang eines Gleises monoton steigend/fallend sind.
- **Duplikat-Erkennung:** Doppelt vergebene Signalbezeichnungen (z.B. zwei Signale mit ID „A102“) werden nicht automatisch erkannt.
- **Fahrtrichtung-Plausibilität:** Es wird nicht validiert, ob die extrahierte Fahrtrichtung mit dem topologischen Streckenverlauf übereinstimmt.
- **Technische Abhängigkeiten:** Regeln wie „Jedes Hauptsignal muss mindestens eine zugeordnete GKS haben“ werden nicht geprüft.

Solche Plausibilitätsprüfungen erfordern domänenspezifisches Expertenwissen und könnten als Erweiterung implementiert werden (vgl. Abschnitt 8.4).

**Fehlende Integration in bestehende Workflows:** Das System ist ein Standalone-Desktop-Tool und nicht in bestehende Siemens-Infrastruktur integriert:

- **Kein CAD-Import:** Direkter Import aus AutoCAD/LCAD ist nicht möglich; Pläne müssen manuell als PDF exportiert werden.
- **Kein Datenbank-Export:** Die Excel-Exporte müssen manuell in Projektdatenbanken (z.B. SAP) importiert werden.
- **Keine Prüfwerkzeug-Anbindung:** Kein automatischer Abgleich mit bestehenden Planprüftools oder Simulationssystemen.
- **Manuelle Datenübertragung:** Copy-Paste oder manuelle Dateiübertragung zwischen Systemen erforderlich.

Eine API-basierte Integration (z.B. REST-API für Planupload und Datenabruf) würde dies adressieren, war jedoch außerhalb des Scopes dieser Masterarbeit.

## 8.4 Verbesserungspotenziale

Basierend auf den identifizierten Limitationen werden konkrete, priorisierte Ansätze zur Systemverbesserung diskutiert. Die Vorschläge sind nach Implementierungsaufwand und erwartetem Nutzen kategorisiert.

### 8.4.1 Kurzfristige Optimierungen (0-6 Monate)

Diese Verbesserungen können mit überschaubarem Aufwand (1-2 Personenwochen pro Maßnahme) implementiert werden und bieten signifikanten, unmittelbaren Nutzen.

#### OCR-Optimierung:

- **Adaptives Upsampling:** Automatische Hochskalierung von Textausschnitten mit Höhe < 40 Pixel vor OCR mittels Bicubic Interpolation. Tests zeigen, dass dies die OCR-Genauigkeit bei kleinen Texten um geschätzt 15-20% steigern könnte.
- **CLAHE-Parameter-Tuning:** Anpassung der Contrast Limited Adaptive Histogram Equalization für bessere Kontraste in schwach belichteten Scans. Aktuell verwendet: Clip-Limit 2.0, Grid-Size 8×8; optimiert könnte 3.0/16×16 bessere Ergebnisse liefern.
- **Post-Processing-Regeln erweitern:** Zusätzliche domänenspezifische Korrekturen:
  - „O“ → „0“ in numerischen Kontexten (z.B. „1O2“ → „102“)
  - „!“ → „1“ bei einzelnen Ziffern
  - Entfernung von Leerzeichen in Signalbezeichnungen („A 102“ → „A102“)

- **EasyOCR standardmäßig aktivieren:** Aktuell ist EasyOCR nur optional; Integration als Standard-Fallback nach Tesseract würde die Robustheit erhöhen. Nachteil: +10% längere Verarbeitungszeit.

### **Linking-Verbesserungen:**

- **Konfidenz-basiertes Scoring:** Gewichtung der Linking-Entscheidung mit YOLO- und OCR-Konfidenz. Beispiel: Bei zwei Kandidaten mit gleicher Distanz sollte der mit höherer OCR-Konfidenz bevorzugt werden.
- **Topologie-Integration:** Berücksichtigung von Gleisverläufen (extrahiert via Hough-Transformation) als zusätzliche Constraint. Beispiel: Koordinaten sollten idealerweise auf der Gleisachse liegen.
- **Konfliktauflösung bei Ambiguität:** Dedizierte Logik für Fälle, in denen mehrere Symbole um einen Text konkurrieren. Aktuell wird nur der nächste Nachbar gewählt; besser wäre eine Gewichtung aus Distanz, Richtung und Konfidenz.

### **Validierung erweitern:**

- **Semantische Plausibilitätsprüfungen:**
  - Kilometrierung-Monotonie: Prüfung, ob Koordinaten entlang eines Gleises monoton steigen/fallen
  - Signal-Duplikat-Erkennung: Warnung bei identischen Signalbezeichnungen
  - Wertebereich-Validierung: Koordinaten sollten im erwarteten Bereich liegen (z.B. 0-200 km für Regionalstrecken)
- **Cross-Element-Validierung:** Prüfung von technischen Abhängigkeiten:
  - Jedes Hauptsignal muss mindestens eine zugeordnete GKS haben
  - GM-Blöcke sollten in definierten Abständen vor Signalen liegen
  - Koordinatenabstände sollten plausibel sein (nicht > 500m Sprung)
- **Statistische Anomalie-Detektion:** Erkennung ungewöhnlicher Muster als Indikator für Fehler:
  - Z-Score-Analyse der Objektdichte (zu viele/wenige Objekte in einem Bereich)
  - Ausreißer-Erkennung bei Rotationswinkel (ungewöhnliche Orientierungen)
  - Abweichung von typischen Text-Längen (zu kurze/lange Bezeichnungen)

**Geschätzte Verbesserung durch kurzfristige Maßnahmen:** End-to-End Accuracy von 97.22% → 99.2-99.5% (Reduktion der Fehlerrate um 35-60%).

### 8.4.2 Mittelfristige Erweiterungen (6-12 Monate)

Diese Maßnahmen erfordern größere Entwicklungsaufwände (1-3 Personenmonate), versprechen jedoch substanzielle Leistungssteigerungen und adressieren fundamentale Limitationen.

#### Neuronales Linking-Modell (Priorität 1):

Dies ist die vom Nutzer priorisierte Verbesserung und hat das größte Potenzial zur Fehlerreduktion.

- **Graph Neural Networks (GNNs):** Modellierung von Symbol-Text-Relationen als Graph-Struktur:
  - *Knoten:* Symbole und Texte als Graphknoten mit Feature-Vektoren (Position, Größe, Klasse, Konfidenz)
  - *Kanten:* Potenzielle Verknüpfungen mit gelernten Gewichten
  - *Architektur:* Graph Convolutional Network (GCN) oder Graph Attention Network (GAT)
  - *Training:* Supervised Learning auf annotierten Symbol-Text-Paaren
- **Attention-basierte Zuordnung:** Transformer-Mechanismus zur Berechnung optimaler Zuordnungen:
  - Self-Attention über alle Symbole und Texte in einem Tile
  - Lernen kontextabhängiger Zuordnungen (z.B. „in Weichenbereichen gelten andere Regeln“)
  - Soft-Attention-Scores statt binärer Entscheidungen (Unsicherheitsquantifizierung)
- **Datenanforderungen:** Geschätzt 50-100 vollständig annotierte Pläne mit expliziten Relation-Labels. Dies entspricht ca. 3000-6000 Symbol-Text-Paaren.
- **Erwarteter Nutzen:** Reduktion der Linking-Fehler von 0.88% auf geschätzt 0.20-0.30%, was die Gesamt-E2E-Accuracy auf 98.5-99.0% heben würde.

#### Datenaugmentation erweitern:

- **Synthetische Layoutvariation:** Generierung künstlicher Pläne mit variierenden Anordnungen:
  - Permutation von Symbolpositionen innerhalb plausibler Bereiche
  - Variation von Textabständen und -orientierungen
  - Simulation unterschiedlicher Objektdichten
- **Style Transfer:** Anwendung von Neural Style Transfer zur Simulation verschiedener Scan-Qualitäten:

- Hinzufügen von Rauschen, Artefakten, Verfärbungen
- Variation von Linienstärken und Schriftarten
- Simulation von Kompressionsartefakten (JPEG-Qualität 50-95)
- **Adversarial Training:** Robustifizierung gegen schwierige Fälle durch gezielt hinzugefügte Störungen:
  - Kleine adversarielle Perturbationen, die YOLO verwirren könnten
  - Training gegen solche Störungen erhöht Robustheit
- **Erwarteter Nutzen:** Bessere Generalisierung auf unbekannte Layouts und Planqualitäten; geschätzt 2-5% höhere Genauigkeit auf neuen Datenquellen.

#### **Active Learning Pipeline:**

- **Unsicherheits-basierte Sampleauswahl:** System identifiziert schwierige Fälle für manuelle Annotation:
  - Objekte mit niedriger YOLO-Konfidenz (< 0.7) automatisch markieren
  - Linking-Entscheidungen mit hoher Ambiguität (mehrere Kandidaten mit ähnlichem Score)
  - OCR-Ergebnisse mit niedriger Engine-Konfidenz oder hoher Inter-Engine-Diskrepanz
- **Inkrementelles Nachtraining:** Periodische Modell-Updates mit neuen Daten aus Produktiveinsatz:
  - Monatliches Retraining mit gesammelten Fehlerfällen
  - Fine-Tuning statt vollständigem Neutrainining (spart Rechenzeit)
  - A/B-Testing neuer Modellversionen gegen Produktivmodell
- **User Feedback Loop:** Korrekturen aus UI fließen zurück in Trainingsdaten:
  - Manuelle Korrekturen werden automatisch als neue Ground-Truth gespeichert
  - Privacy-Schutz: Nur mit Nutzereinwilligung
  - Versionierung: Nachvollziehbarkeit, welche Daten wann hinzugefügt wurden
- **Erwarteter Nutzen:** Kontinuierliche Verbesserung der Modellqualität über Lebenszeit; geschätzt 1-2% Genauigkeitssteigerung pro Jahr.

#### **GPU-Beschleunigung und Optimierung:**

- **Batch-Inferenz:** Parallele Verarbeitung mehrerer Tiles auf GPU:
  - Aktuell: Sequenzielle Verarbeitung von Tiles

- Optimiert: Batch von 8-16 Tiles gleichzeitig
- Geschätzte Beschleunigung: 5-10× (10.6 min → 1.2-2.5 min pro Plan)
- **Model Optimization:** Quantisierung und Pruning für schnellere Inferenz:
  - INT8-Quantisierung statt FP32 (4× kleineres Modell, 2-3× schneller)
  - Structured Pruning: Entfernung unwichtiger Netzwerk-Connections
  - Knowledge Distillation: Training eines kleineren „Student“-Modells
- **Hybrid CPU/GPU:** Kritische Pfade (YOLO, OCR) auf GPU, Rest auf CPU:
  - Automatische Fallback-Logik: Falls GPU nicht verfügbar, CPU-Modus
  - Optimale Ressourcennutzung auf heterogenen Systemen
- **Training bei nativer Auflösung:** Das aktuelle Modell wurde auf  $1024 \times 1024$  Pixel trainiert (Downsampling von  $2048 \times 2048$  Annotationen), um Trainingszeit und GPU-Speicherbedarf zu reduzieren. Mit leistungsfähigerer GPU-Infrastruktur (z.B. NVIDIA A100 mit 80 GB VRAM oder Multi-GPU-Cluster) könnte das Modell direkt auf  $2048 \times 2048$  Pixeln trainiert werden:
  - Erhalt feinerer visueller Details ohne Downsampling-Verluste
  - Potenziell verbesserte Erkennung kleiner Symbole (Isolierstöße, GKS)
  - Cloud-basierte GPU-Instanzen (AWS p4d, Azure NC-Series) für einmaliges Retraining nutzbar
- **Erwarteter Nutzen:** Reduktion der Verarbeitungszeit auf < 3 Minuten pro Plan bei GPU-Verfügbarkeit; verbesserte Skalierbarkeit für Batch-Verarbeitung. Training bei nativer Auflösung könnte zusätzlich die Detektionsgenauigkeit bei kleinen Symbolen verbessern.

**Geschätzte Verbesserung durch mittelfristige Maßnahmen:** End-to-End Accuracy von 97.22% → 99.3-99.6%; Verarbeitungszeit von 10.6 min → 2-3 min (GPU) oder 10 min (optimiertes CPU).

### 8.4.3 Langfristige Forschungsrichtungen (12+ Monate)

Diese Ansätze adressieren fundamentale Limitationen, erfordern substanziale Forschung (6-12 Personenmonate) und sind teilweise noch Gegenstand aktiver wissenschaftlicher Forschung.

#### Multimodales End-to-End-Lernen:

- **Vision-Language Models (VLM):** Integration von Text- und Bildverständnis in einem Modell:
  - Modelle wie CLIP, LLaVA oder Flamingo als Basis

- Fine-Tuning auf technischen Zeichnungen mit Text-Bild-Paaren
- Direktes Lernen: Bild → strukturierter Output ohne separate OCR/Linking-Stufen
- **Layout Understanding:** Explizite Modellierung von Dokumentstruktur:
  - Hierarchische Repräsentation: Plan → Bereiche → Symbole → Attribute
  - Transformer-basierte Layout-Encoder (wie in LayoutLMv3)
  - Lernen räumlicher Relationen („unterhalb“, „neben“, „zwischen“)
- **Kontextuelles Reasoning:** Berücksichtigung globaler Plankonsistenz:
  - „Dieses Symbol ist wahrscheinlich ein Signal, weil in der Nähe eine GKS ist“
  - „Diese Kilometrierung ist plausibel, weil sie zwischen zwei bekannten Werten liegt“
  - Graph-Attention über den gesamten Plan statt tile-lokal Verarbeitung
- **Herausforderungen:**
  - Benötigt massive Datenmengen (1000+ annotierte Pläne)
  - Hoher Rechenaufwand (Multi-GPU-Cluster für Training)
  - Interpretierbarkeit reduziert (Black-Box-Problem)
- **Erwarteter Nutzen:** Potenzial für 99.5-99.8% E2E-Accuracy durch Vermeidung von Kaskaden-Fehlern; jedoch unsicher und hochrisikoreich.

### **Transfer Learning und Domänenadaption:**

- **Cross-Domain Pre-Training:** Lernen von ähnlichen Domänen für Robustheit:
  - Pre-Training auf P&ID-Diagrammen (größere verfügbare Datensätze)
  - Transfer auf Gleispläne durch Fine-Tuning
  - Gemeinsame visuelle Muster nutzen (Symbole, Linien, Texte)
- **Few-Shot Learning:** Schnelle Anpassung an neue Symboltypen mit wenigen Beispielen:
  - Meta-Learning-Ansätze (z.B. MAML, Prototypical Networks)
  - „Lernen zu lernen“: Modell kann sich mit 5-10 Beispielen an neue Klasse anpassen
  - Reduktion des Annotationsaufwands für neue Layouts
- **Domain Randomization:** Training auf maximal diversifizierten Daten:
  - Extreme Variation von Farben, Kontrasten, Verzerrungen
  - Modell lernt invariante Features statt spezifischer Artefakte
  - Bessere Generalisierung auf unbekannte Datenquellen

- **Erwarteter Nutzen:** Reduktion des Datenanforderungen für neue Layouts um 50-70%; robustere Generalisierung auf unbekannte Quellen.

### Explainable AI und Vertrauenswürdigkeit:

- **Attention Visualization:** Visuelle Erklärungen von Modellentscheidungen:
  - Grad-CAM oder SHAP für YOLO: Welche Bildbereiche führten zur Detektion?
  - OCR-Attention: Welche Pixel wurden für welches Zeichen verwendet?
  - Linking-Explanation: Warum wurde diese Text-Symbol-Zuordnung getroffen?
- **Confidence Calibration:** Realistische Unsicherheitsabschätzungen:
  - Aktuell: YOLO-Konfidenz korreliert nicht perfekt mit tatsächlicher Korrektheit
  - Kalibrierung durch Temperature Scaling oder Platt Scaling
  - Bessere Priorisierung manueller Prüfung (niedrige Konfidenz = hohes Fehlerrisiko)
- **Counterfactual Explanations:** „Was-wäre-wenn“-Analysen:
  - „Wenn dieser Text 50 Pixel weiter rechts wäre, würde er dem anderen Symbol zugeordnet“
  - Hilft Nutzern, Systemlogik zu verstehen und Fehlerquellen zu identifizieren
- **Erwarteter Nutzen:** Höhere Nutzerakzeptanz durch Transparenz; verbesserte Fehlerdiagnose und -korrektur.

### Integration in digitale Zwillinge:

- **3D-Rekonstruktion:** Transformation 2D-Plan zu 3D-Modell:
  - Extraktion der Gleistopologie als 3D-Graph
  - Positionierung von Signalen, Weichen etc. im 3D-Raum
  - Kombination mit Höhenprofilen und Geländedaten
- **Simulation Integration:** Direkte Nutzung extrahierter Daten für Betriebssimulationen:
  - Export in Simulationsformate (OpenTrack, RailSys)
  - Automatisierte Fahrplan-Validierung
  - Kapazitätsanalysen und Engpass-Identifikation
- **Realtime Monitoring:** Abgleich Plan-Daten mit Sensor-Daten:
  - Vergleich Soll-Zustand (Plan) mit Ist-Zustand (Sensoren)
  - Erkennung von Abweichungen (fehlende/zusätzliche Elemente)
  - Automatisierte Planaktualisierung bei Infrastrukturänderungen

- **Erwarteter Nutzen:** Nahtlose Integration in digitale Infrastruktur-Verwaltung; Grundlage für Predictive Maintenance und automatisierte Betriebsführung.

## 8.5 Generalisierbarkeit und Übertragbarkeit

Ein zentrales Ziel dieser Arbeit war die Entwicklung eines Ansatzes, der prinzipiell auf andere technische Zeichnungsdomänen übertragbar ist. Während die konkrete Implementierung auf Gleispläne spezialisiert ist, sind die zugrundeliegenden Konzepte und Komponenten domänenübergreifend anwendbar.

### 8.5.1 Übertragbarkeit auf verwandte Domänen

Die entwickelte Pipeline-Architektur ist konzeptionell nicht auf Gleispläne beschränkt. Folgende Domänen könnten mit ähnlichen Ansätzen adressiert werden:

#### Piping & Instrumentation Diagrams (P&ID):

- **Ähnlichkeiten:** Standardisierte Symbole für Komponenten (Ventile, Pumpen, Messgeräte), rotierte Orientierungen entlang Rohrleitungen, Text-Symbol-Beziehungen (Tag-Numbers, Spezifikationen)
- **Unterschiede:** Komplexere Topologie mit vielen Verbindungslien und Kreuzungen; hierarchische Verschachtelung (Haupt-/Nebenleitungen); mehr Textannotationen (Drücke, Temperaturen, Durchflüsse)
- **Anpassungen:**
  - Training auf P&ID-Symboldatensatz (geschätzt 30-50 Klassen)
  - Erweiterte Linking-Heuristiken für Rohrleitungsverfolgung (Line-Tracing-Algorithmus)
  - OCR-Validierung für numerische Werte mit Einheiten (z.B. „150 bar“, „80°C“)
- **Geschätzter Aufwand:** 2-3 Personenmonate für Anpassung und 50-100 annotierte Pläne

#### Elektrische Installationspläne:

- **Ähnlichkeiten:** Klar definierte Symbolbibliothek nach DIN/IEC-Normen, Koordinaten und Bezeichnungen (Raumnummern, Stromkreis-IDs), rotationsinvariante Komponenten
- **Unterschiede:** Hierarchische Struktur (Hauptverteiler → Unterverteiler → Endstromkreise), tabellarische Legenden mit vielen Textfeldern, farbkodierte Leitungen (RGB-Information relevant)
- **Anpassungen:**
  - Hierarchisches Linking (Zuordnung von Geräten zu Stromkreisen zu Verteilern)
  - Erweiterte OCR für Tabellen und Legenden (Table-Detection-Modul)

- Farbkanalnutzung statt Grauwert-Verarbeitung
- **Geschätzter Aufwand:** 3-4 Personenmonate für Anpassung und 60-80 annotierte Pläne

#### Hydraulikschaltpläne:

- **Ähnlichkeiten:** Fluss-basierte Darstellung mit standardisierten Komponenten (ISO 1219), Druckangaben und Durchflussraten als Text, orientierte Symbole (Strömungsrichtung)
- **Unterschiede:** Komplexe Ventilsymbole mit internen Details (Schaltstellungen), simultane Darstellung mehrerer Systemzustände, enge Symbol-Platzierung
- **Anpassungen:**
  - Feinkörnigere Symbolklassifikation (z.B. 4/3-Wegeventil vs. 5/2-Wegeventil)
  - OCR für Druckwerte und Einheiten als zusätzlicher Texttyp
  - Multi-State-Handling (Erkennung verschiedener Schaltstellungen)
- **Geschätzter Aufwand:** 2-3 Personenmonate für Anpassung und 40-60 annotierte Pläne

#### 8.5.2 Generalisierbare Komponenten und Muster

Mehrere Aspekte des Prototyps haben domänenübergreifende Relevanz und können direkt wiederverwendet werden:

##### Technische Muster:

- **OBB-basierte Objekterkennung:** Universell anwendbar für rotierte Symbole in jeder technischen Zeichnung. Die Entscheidung für YOLOv8-OBB statt HBB ist für alle Domänen mit nicht-achsenparallelen Objekten vorteilhaft.
- **Synthetische Rotationsaugmentation:** Effektiv für geometrische Varianz bei kleinen Datensätzen. Der Ansatz, Trainingsdaten durch 10 Rotationsschritte zu verzehnfachen, ist auf P&ID, Elektropläne etc. direkt übertragbar.
- **Multi-Engine OCR-Kaskadierung:** Robustifizierung durch Diversität der Engines (PaddleOCR → Tesseract → EasyOCR) funktioniert universell für technischen Text. Die Fallback-Strategie ist domänenunabhängig.
- **Orientierungsadaptive Preprocessing (Dual-Path-Routing):** Die Idee, Textausschnitte in zwei Orientierungen (Original + 90°) zu verarbeiten, ist auf jede Zeichnung mit rotiertem Text anwendbar.

##### Architekturmuster:

- **Modulare Pipeline:** Die klare Trennung YOLO → OCR → Linking → Validierung ermöglicht komponentenweisen Austausch. Für eine neue Domäne können z.B. Linking-Regeln angepasst werden, ohne YOLO/OCR zu ändern.

- **Regelbasiert + Lernbasiert hybrid:** Pragmatischer Ansatz für begrenzte Datenmengen. Die Kombination von geometrischen Heuristiken (Regeln) mit adaptivem Lernen (Mustererkennung) ist universell auf Domänen mit < 100 Trainingsplänen anwendbar.
- **Validierung mit Fuzzy-Matching:** Domänen spezifische Fehlerkorrektur durch Regex-Validierung und Plausibilitätsprüfungen. Das Konzept der dreistufigen Validierung (Symbol → OCR → Linking) ist übertragbar.
- **Human-in-the-Loop UI:** Effiziente Qualitätssicherung durch visuelle Werkzeuge (Jump-to-Detection, Inline-Editing). Die PyQt5-basierte UI ist mit minimalen Anpassungen für andere Zeichnungstypen nutzbar.

#### **Prozessmuster:**

- **Tiling-Strategie:** Notwendig bei großformatigen Plänen (A0/A1). Die Wahl von 2048×2048 Pixeln mit 12.5% Überlappung ist ein guter Ausgangspunkt für andere Domänen.
- **Ground-Truth-Erstellung mit CVAT:** Standardisierter Workflow für OBB-Annotation. Das CVAT-Tool ist domänenunabhängig und für jede Objekterkennung verwendbar.
- **Iterative Evaluation:** Systematische Metrik-Erhebung über Validierungs- und Testsätze. Die Verwendung von mAP, Recall, Precision für Detection und Field Accuracy für E2E ist Standard und übertragbar.

### **8.5.3 Nicht-übertragbare domänenspezifische Aspekte**

Einige Komponenten sind eng an die Gleisplan-Domäne gekoppelt und erfordern bei Übertragung substantielle Anpassungen:

- **Linking-Heuristiken:** Die Richtungspräferenzen (z.B. „Koordinate unterhalb von Symbol“, „Signalbezeichnung rechts von Signal“) basieren auf Gleisplan-Konventionen. Für P&ID oder Elektropläne gelten andere Regeln (z.B. Tag-Numbers oft oberhalb von Equipment).
- **Fahrrichtungsdetektion:** Die geometrische Ableitung der Fahrrichtung aus der relativen Position von Signal und GKS (festkodiert) ist eisenbahnspezifisch. Andere Domänen haben keine äquivalente Konzepte oder benötigen andere Logiken (z.B. Strömungsrichtung in Hydraulikplänen aus Pfeilsymbolen).
- **Regex-Validierung:** Formatmuster für Signalbezeichnungen (z.B. texttt[A-Z]{1,3}[0-9]+) und Koordinaten (z.B. texttt  
textbackslash d+[.,]  
textbackslash d+) müssen neu definiert werden. P&ID-Tag-Numbers folgen z.B. dem Schema textttXX-NNNN-SSS (Equipment-Type, Nummer, Suffix).

- **Symbolklassen:** Die 13 Klassen (Signal, Koordinate, GKS, GM-Block, Weiche, ...) müssen komplett neu definiert werden. P&ID hätte z.B. Ventile, Pumpen, Tanks, Messgeräte; Elektropläne hätten Schalter, Steckdosen, Leuchten, Verteiler.
- **Auflösungsanforderung:** Die 500 DPI Anforderung ist spezifisch für das getestete Gleisplan-Layout. Andere Domänen mit größeren Symbolen (z.B. Architekturpläne) könnten mit 300 DPI auskommen; feinere Details (z.B. Mikroelektronik-Schaltpläne) könnten 1000+ DPI erfordern.

**Fazit zur Übertragbarkeit:** Etwa 60-70% des Prototyps (Architektur, YOLO-Training, OCR-Pipeline, UI) sind mit moderatem Aufwand übertragbar. Die verbleibenden 30-40% (Linking-Logik, Validierungsregeln, Symbolklassen) erfordern domänen spezifische Anpassungen und Re-Training. Dies ist wesentlich effizienter als eine Neuentwicklung von Grund auf, die 100% Aufwand bedeuten würde.

## 8.6 Ausblick und zukünftige Entwicklungen

Abschließend wird ein Ausblick auf zukünftige Entwicklungen gegeben, die sowohl das konkrete System bei Siemens Mobility als auch das breitere Forschungsfeld der automatisierten Dokumentenverarbeitung betreffen.

### 8.6.1 Evolution des konkreten Systems

#### Geplante Entwicklungsschritte bei Siemens Mobility:

Die erfolgreiche Evaluation (97.22% E2E-Accuracy, 89,7% Zeitersparnis, vgl. Kapitel 7) hat zu internen Diskussionen über den Produktiveinsatz geführt. Ein möglicher Rollout-Plan könnte folgendermaßen aussehen:

1. **Pilotphase (Q2 2026):** Einsatz in 2-3 ausgewählten Projekten mit intensivem Monitoring:
  - Wöchentliche Erfassung von Fehlerfällen und User-Feedback
  - Qualitative Interviews mit Ingenieuren zur Usability
  - Quantitative Messung der tatsächlichen Zeitersparnis in operativer Umgebung
  - Identifikation von Edge-Cases und systematischen Fehlern
2. **Datensammlung (Q3 2026):** Systematische Erfassung von Korrekturen:
  - Automatisches Logging aller manuellen Korrekturen über die UI
  - Aufbau eines User-Feedback-Datasets (geschätzt 50-100 zusätzliche Pläne)
  - Kategorisierung von Fehlertypen für priorisierte Verbesserungen
3. **Modell-Update (Q4 2026):** Nachtraining mit gesammelten Daten:

- Fine-Tuning des YOLO-Modells mit korrigierten Annotationen
- Erweiterung der OCR-Validierungsregeln basierend auf beobachteten Fehlermustern
- Update der Linking-Heuristiken mit neu identifizierten Layoutmustern
- Ziel: Steigerung der E2E-Accuracy von 97.22% auf > 99%

#### 4. Produktivbetrieb (2027): Rollout für breitere Anwenderbasis:

- Deployment auf Standard-Siemens-Workstations
- Integration in bestehende CAD-Workflows (z.B. AutoCAD-Plugin)
- Schulungen für ca. 50-100 Ingenieure
- Etablierung eines Support-Prozesses für Fehlerberichte

#### Mögliche Erweiterungen:

Über die Basisversion hinaus werden folgende Features diskutiert:

- **Zusätzliche Symbolklassen:** Sukzessive Erweiterung auf weitere Bahnelemente:
  - Priorität 1: LZB-Komponenten (Linienleiter, Übertragungseinrichtungen)
  - Priorität 2: Bahnübergangstechnik (Schranken, Lichtzeichen)
  - Priorität 3: Energieversorgung (Kabelverläufe, Verteilerkästen)
- **Multi-Format-Support:** Direkter Import aus CAD-Systemen:
  - DWG/DXF-Parsing für direkte Vektordaten (umgeht PDF-Konvertierung)
  - Erhalt von Layer-Informationen (z.B. separate Layer für Signale, Gleise, Text)
  - Potenzielle Genauigkeitssteigerung durch Nutzung strukturierter CAD-Daten
- **API-Integration:** REST-API für Systemintegration:
  - Upload-Endpoint: POST /plans mit PDF-Datei
  - Status-Endpoint: GET /plans/{id}/status für Fortschrittsanzeige
  - Results-Endpoint: GET /plans/{id}/results für JSON/Excel-Export
  - Ermöglicht Integration in Projektmanagement-Systeme und Datenbanken
- **Cloud-Deployment (optional):** Für Performance-kritische Szenarien:
  - Azure/AWS-basierte GPU-Instanzen für Batch-Verarbeitung
  - Reduktion der Verarbeitungszeit auf < 1 Minute pro Plan
  - Strenge Datenschutzkontrollen (Verschlüsselung, DSGVO-Konformität)
  - Nur für unkritische Daten ohne Geheimhaltungsstufe

**Upgrade-Fähigkeit der Detektionskomponente:**

Die modulare Architektur ermöglicht den Austausch des YOLO-Modells mit geringem Aufwand. Ein Upgrade auf neuere Versionen (z.B. YOLOv9, YOLOv10, YOLOv11) erfordert lediglich:

1. Modell-Neutrainung mit den bestehenden Annotationen (das YOLO-OBB-Format ist versionsübergreifend kompatibel)
2. Anpassung des Modellpfads in der Konfigurationsdatei
3. Validierung auf dem unveränderten Testsatz zur Sicherstellung der Vergleichbarkeit

Der geschätzte Gesamtaufwand beträgt 2-4 Personentage, da OCR-, Linking- und Validierungs-komponenten unverändert bleiben.

Für systematische Versionsvergleiche sollte die Evaluation auf identischem Testsatz (7 Produktionspläne) mit konsistenten Metriken (mAP@0.5, E2E-Accuracy, Inferenzzeit) unter gleichen Hardwarebedingungen erfolgen. Basierend auf publizierten Benchmarks der YOLO-Familie ist bei neueren Versionen eine moderate Genauigkeitssteigerung (1-3% mAP) sowie eine Inferenzbeschleunigung (10-30%) zu erwarten. Da das aktuelle System bereits 98.4% mAP erreicht, liegt das Optimierungspotential primär in der Verarbeitungsgeschwindigkeit und der Robustheit bei Grenzfällen.

### 8.6.2 Trends in der automatisierten Dokumentenverarbeitung

**Foundation Models für Document AI:**

Die rasante Entwicklung großer multimodaler Modelle (GPT-4 Vision, Gemini 1.5, Claude 3) deutet auf eine Zukunft hin, in der generische Document Understanding Modelle auch spezialisierte Domänen abdecken könnten. Diese Modelle zeigen beeindruckende Fähigkeiten bei der Interpretation von Diagrammen, Tabellen und technischen Zeichnungen durch natürlichsprachliche Prompts.

Jedoch bleibt offen, ob diese Modelle die hier erreichte Präzision für hochspezialisierte technische Zeichnungen erreichen werden:

- **Genauigkeitsanforderungen:** 97.22% E2E-Accuracy erfordert nahezu perfekte Fehlervermeidung. Foundation Models neigen zu „Halluzinationen“ und inkonsistenten Outputs.
- **Sicherheitskritische Anwendungen:** Bei Bahnanlagen sind Fehler potentiell lebensgefährlich. Die Nachvollziehbarkeit und Reproduzierbarkeit von Foundation Model Outputs ist problematisch.
- **On-Premise-Anforderung:** Große Modelle (100B+ Parameter) erfordern Cloud-Infrastruktur. Die Anforderung NFA-001 (lokale Verarbeitung) würde verletzt.
- **Kosten:** Inference-Kosten für Foundation Models (z.B. \$0.01-0.05 pro Bild) wären bei Hunderten Plänen prohibitiv im Vergleich zu einmaligen Trainingskosten.

Dennoch könnten Foundation Models als *Ergänzung* wertvoll sein, z.B. für:

- Automatisierte Generierung von Validierungsregeln aus Beispielen
- Zero-Shot-Erkennung neuer Symboltypen ohne Training
- Natürlichsprachliche Abfragen über extrahierte Daten („Zeige alle Signale zwischen km 12 und km 15“)

### Synthetic Data Generation:

Fortschritte in generativen Modellen (Diffusion Models, GANs, insbesondere ControlNet und SDXL) ermöglichen zunehmend die Synthese realistischer technischer Zeichnungen. Dies könnte die Herausforderung begrenzter Trainingsdaten fundamental adressieren:

- **Layout-Generierung:** Automatische Erzeugung von Gleisplan-Varianten mit kontrollierbaren Parametern (Objektdichte, Komplexität, Layout-Typ)
- **Symbol-Variation:** Generierung leicht varierter Symbol-Designs zur Erhöhung der Trainingsvielfalt
- **Qualitäts-Simulation:** Synthetische Degradation (Rauschen, Artefakte, Kompression) zur Robustifizierung

Erste Forschungsarbeiten in diesem Bereich (z.B. „Synthetic P&ID Generation via Diffusion Models“) zeigen vielversprechende Ergebnisse. Für Gleispläne existieren jedoch noch keine vergleichbaren Ansätze – eine mögliche Forschungsrichtung.

### Neuro-Symbolic AI:

Die Integration neuronaler Komponenten mit symbolischer Regelverarbeitung wird zunehmend als vielversprechender Ansatz erkannt. Diese Arbeit implementiert ansatzweise einen solchen Hybrid-Ansatz:

- **Neuronale:** YOLO für Objekterkennung, CNN-basierte OCR
- **Symbolisch:** Regex-Validierung, Linking-Heuristiken, Plausibilitätsprüfungen

Zukünftige Systeme könnten dies vertiefen durch:

- **Ontologien technischer Zeichnungen:** Formale Wissensrepräsentation (z.B. in OWL) von Gleisplan-Konventionen und -Regeln
- **Logisches Reasoning:** Automatische Ableitung von Implikationen („Wenn Signal A102 existiert, muss es eine GKS geben“)
- **Constraint Solving:** Auflösung von Ambiguitäten durch Constraint Satisfaction Problems (CSP)

Diese Kombination könnte die Interpretierbarkeit neuronaler Systeme erhöhen und gleichzeitig die Flexibilität symbolischer Systeme steigern.

### 8.6.3 Gesellschaftliche und ethische Implikationen

Die zunehmende Automatisierung der Dokumentenverarbeitung wirft auch breitere gesellschaftliche Fragen auf:

#### Arbeitsmarkt und Qualifikationen:

Die durch dieses System erzielte Zeitersparnis von 89,7% (vgl. Tabelle 7.17) bedeutet, dass statt 102 Minuten nur noch 10,6 Minuten menschliche Arbeit pro Plan benötigt werden. Dies hat Auswirkungen auf:

- **Tätigkeitsprofile:** Weniger repetitive Extraktionsarbeit, mehr Fokus auf Validierung, Plausibilitätsprüfung und Engineering-Entscheidungen
- **Qualifikationsanforderungen:** Ingenieure benötigen Verständnis von KI-Systemen und deren Limitationen („AI Literacy“)
- **Produktivitätsgewinne:** Mehr Projekte können mit gleichen Personalressourcen bearbeitet werden, was wirtschaftlich vorteilhaft ist
- **Potenzielle Risiken:** Langfristig könnte hochgradige Automatisierung zu Stellenabbau führen, jedoch ist dies bei sicherheitskritischen Anwendungen unwahrscheinlich (menschliche Expertise bleibt essentiell)

Diese Transformation erfordert organisatorische Anpassungen: Weiterbildungsprogramme, neue Prozessedefinitionen, Anpassung von Stellenbeschreibungen.

#### Verantwortung und Haftung:

Bei sicherheitskritischen Anwendungen stellt sich die fundamentale Frage der Verantwortlichkeit:

- **Fehlerszenarien:** Was passiert, wenn eine fehlerhafte KI-Extraktion zu einem Planungsfehler führt, der später zu einem Sicherheitsvorfall beiträgt?
- **Haftungskette:** Wer trägt die Verantwortung?
  - Der Ingenieur, der die KI-Ergebnisse validiert hat?
  - Der Entwickler des KI-Systems?
  - Siemens Mobility als Betreiber der Software?
  - Der Auftraggeber, der das System einsetzt?
- **Rechtslage:** Die aktuelle Rechtsprechung (z.B. EU AI Act) ist noch nicht vollständig geklärt bezüglich KI in sicherheitskritischen Anwendungen
- **Empfehlung:** Klare Prozessdefinition mit eindeutiger Zuordnung: KI als *Werkzeug*, Ingenieur als *verantwortlicher Entscheider*

### Transparenz und Nachvollziehbarkeit:

Gerade im regulierten Eisenbahnbereich (EBO, TSI-Normen, CENELEC-Standards) ist die Nachvollziehbarkeit von Entscheidungen essentiell:

- **Dokumentationspflichten:** Alle Planungsschritte müssen revisionssicher dokumentiert werden
- **KI-Nachvollziehbarkeit:** Das System muss erklären können, warum eine bestimmte Extraktion vorgenommen wurde
- **Audit-Fähigkeit:** Externe Prüfer (z.B. Eisenbahn-Bundesamt) müssen Systementscheidungen nachvollziehen können

Die in dieser Arbeit gewählte modulare Architektur mit expliziten Validierungsstufen adressiert dies besser als Black-Box-End-to-End-Modelle, jedoch bleibt Verbesserungspotenzial (vgl. Explainable AI in Abschnitt 8.4).

## 8.7 Beantwortung der Forschungsfragen

Die in Kapitel 2 formulierten Forschungsfragen können auf Basis der durchgeführten Implementierung und Evaluation wie folgt beantwortet werden.

### 8.7.1 Hauptforschungsfrage (HF)

*Wie lässt sich der Prozess der Datenextraktion aus heterogenen technischen Zeichnungen durch den Einsatz von Deep Learning und hybriden Verarbeitungsstrategien automatisieren, um eine valide Überführung in strukturierte Datenmodelle zu gewährleisten?*

**Antwort:** Die vorliegende Arbeit demonstriert, dass eine erfolgreiche Automatisierung durch die Kombination dreier Kernkomponenten erreicht werden kann:

1. **Deep Learning für Objekterkennung:** YOLOv8 mit Oriented Bounding Boxes (OBB) ermöglicht die rotationsinvariante Detektion domänenpezifischer Symbole mit einer mAP@0.5 von 98.4% (Tabelle 7.4).
2. **Hybride OCR-Strategie:** Die Multi-Engine-Kaskade (PaddleOCR → Tesseract → EasyOCR) mit orientierungsadaptiver Vorverarbeitung (Dual-Angle-Routing) erreicht eine implizite OCR-Genauigkeit von 98.17% (27 von 1473 Objekten mit OCR-Fehlern).
3. **Regelbasiertes Linking:** Räumliche Verknüpfungsalgorithmen mit lokaler Koordinatentransformation (Abschnitt 6.4.1) assoziieren Symbole und Texte mit hoher Genauigkeit.

Die End-to-End-Genauigkeit von **97.22%** auf dem Testsatz (Tabelle 7.7) belegt die Validität der Überführung in strukturierte Datenmodelle. Die PostgreSQL-Persistierung mit JSONB-Speicherung gewährleistet die strukturierte Datenhaltung, während der Excel-Export die Integration in nachgelagerte Planungsprozesse ermöglicht.

### 8.7.2 Teilstudienfrage 1 (TF1)

*Inwieweit eignen sich aktuelle einstufige Objektdetektoren zur zuverlässigen Erkennung von kleinteiligen, rotierten Symbolen in technischen Zeichnungen und welche Vorverarbeitungsschritte sind notwendig, um die Präzision bei variierenden Layouts zu maximieren?*

**Antwort:** Einstufige Objektdetektoren – konkret YOLOv8 in der OBB-Variante – eignen sich hervorragend für diese Aufgabe, sofern spezifische Vorverarbeitungsschritte implementiert werden:

- **Tiling-Strategie:** Die Zerlegung großformatiger A0-Pläne in überlappende Tiles (2048×2048 Pixel, 12.5% Overlap, 128px Halo) ermöglicht die Verarbeitung bei gleichzeitiger Erhaltung der Symboldetails (Abschnitt 6.2.3).
- **Hochauflösende Rasterisierung:** Die Rendering-Auflösung von 500 DPI ist zwingend erforderlich, um filigrane Symbole differenzierbar darzustellen (Kapitel 5).
- **Synthetische Rotationsaugmentation:** Die Erweiterung des Trainingsdatensatzes durch 10 Rotationswinkel ( $\pm 15^\circ$ ,  $\pm 30^\circ$ ,  $\pm 45^\circ$ ,  $\pm 60^\circ$ ,  $\pm 90^\circ$ ) kompensiert die begrenzte Datensetze effektiv und führt zu vollständiger Rotationsinvarianz. Bemerkenswert ist, dass rotierte Objekte ( $|\theta| > 30^\circ$ ) sogar höhere Konfidenzwerte aufweisen (0.946 vs. 0.890) als horizontal ausgerichtete (Tabelle 7.9).
- **Klassengewichtete Augmentation:** Unterrepräsentierte Klassen (z.B. *endeweichen* mit nur 4 Original-Instanzen) wurden durch höhere Augmentationsfaktoren (bis zu 150×) ausgeglichen.

Die erreichten Metriken (Precision: 97.5%, Recall: 95.7%, mAP@0.5: 98.4%, vgl. Tabelle 7.4) übertreffen vergleichbare Arbeiten zur P&ID-Erkennung (typisch 85-90%) und validieren die Eignung einstufiger Detektoren für diese Domäne.

### 8.7.3 Teilstudienfrage 2 (TF2)

*Wie können geometrische Informationen und unstrukturierte Textdaten algorithmisch so verknüpft werden, dass eine korrekte semantische Zuordnung (z. B. Signalbezeichnung zu Signalsymbol) auch bei hoher Objektdichte erfolgt?*

**Antwort:** Die algorithmische Verknüpfung erfolgt durch einen mehrstufigen Ansatz, der geometrische Nähe mit domänenspezifischen Regeln kombiniert:

1. **Rotationsinvariante Koordinatentransformation:** Für jedes Ankersymbol wird ein lokales Koordinatensystem etabliert, dessen Achsen mit der Symbolorientierung übereinstimmen. Dies ermöglicht die Definition relativer Positionen (“oberhalb”, “unterhalb”) unabhängig von der absoluten Symbolrotation (Abschnitt 6.4.1).
2. **Klassenspezifische Suchbereiche:** Für jede Symbolklasse werden empirisch kalibrierte

Suchparameter definiert ( $dy_{max}$ ,  $dx_{max}$ ), die typische Abstände zwischen Symbolen und zugehörigen Beschriftungen bei 500 DPI abbilden (Tabelle 6.11).

3. **Richtungsbasierte Filterung:** Die erwartete Textposition relativ zum Symbol wird klassenspezifisch definiert (z.B. Koordinaten unterhalb von Signalen), wodurch falsche Kandidaten bei hoher Objektdichte ausgeschlossen werden.
4. **Regex-basierte Validierung:** Extrahierte Texte werden gegen domänen spezifische Muster geprüft (Signal: [A-Z]{1,4}\d{1,4}, Koordinate: \d+[.,]\d+), um fehlerhafte Zuordnungen zu erkennen.

Mit 13 Linking-Fehlern bei 1473 Objekten (0.88% Fehlerrate) demonstriert dieser Ansatz, dass auch bei hoher Objektdichte – die Testpläne enthielten durchschnittlich 164 Objekte pro Plan – eine zuverlässige semantische Zuordnung möglich ist.

#### 8.7.4 Teilforschungsfrage 3 (TF3)

*Wie muss eine Systemarchitektur gestaltet sein, um neue Symbolklassen und Datenformate modular zu integrieren, wobei Änderungen auf einzelne Pipeline-Komponenten beschränkt bleiben?*

**Antwort:** Die implementierte Architektur folgt dem Prinzip der *Separation of Concerns* mit klar definierten Schnittstellen zwischen Komponenten:

- **Modulare Pipeline-Struktur:** Die Verarbeitungskette (PDF-Rasterisierung → Tiling → YOLO-Inferenz → OCR → Linking → Validierung → Export) ist in eigenständige Module gekapselt, die über definierte Datenstrukturen kommunizieren.
- **Klassenspezifische Parametrisierung:** Linking-Parameter, OCR-Vorverarbeitung und Validierungsregeln sind klassenspezifisch konfiguriert (Tabellen 6.11, 6.12), sodass neue Symbolklassen durch Hinzufügen entsprechender Parameter integriert werden können.
- **Isolierte Modellkomponente:** Das YOLO-Modell ist austauschbar; ein Nachtraining für neue Symbolklassen erfordert keine Änderungen an OCR-, Linking- oder Export-Logik.
- **Erweiterbare Exportschicht:** Die Trennung von Datenextraktion und -export ermöglicht die Hinzufügung neuer Ausgabeformate ohne Modifikation der Kernpipeline.

Die erfolgreiche Integration aller 13 Symbolklassen validiert dieses Design. Die Hinzufügung neuer Klassen erfordert lediglich: (1) Annotation und Training des YOLO-Modells, (2) Definition klassenspezifischer Linking-Parameter, (3) Hinzufügung von Regex-Mustern für die Textvalidierung.

#### 8.7.5 Teilforschungsfrage 4 (TF4)

*Welcher algorithmische Ansatz eignet sich, um in rein visuellen Daten semantische Änderungen zwischen zwei Planversionen robust zu identifizieren und visualisierbar zu machen?*

**Antwort:** Der implementierte Ansatz basiert auf dem *Vergleich strukturierter Extraktionsergebnisse* anstatt auf direktem Bildvergleich:

1. **Extraktion beider Versionen:** Sowohl die alte als auch die neue Planversion durchlaufen die vollständige Extraktionspipeline, wodurch strukturierte Datensätze (Symbol-ID, Position, Bezeichnung, Attribute) entstehen.
2. **Semantischer Abgleich:** Der Matching-Prozess erfolgt zweistufig: Zunächst werden Objekte anhand eindeutiger Kennungen (UUID) direkt zugeordnet. Für nicht-zugeordnete Elemente wird der *Hungarian Algorithm* (Kuhn-Munkres-Algorithmus) eingesetzt, der eine optimale Zuordnung auf Basis einer Kostenmatrix berechnet. Diese berücksichtigt sowohl die räumliche Distanz (coord\_value) als auch die Ähnlichkeit der OCR-Texte.
3. **Kategorisierung der Änderungen:** Identifizierte Differenzen werden klassifiziert als:
  - *Hinzugefügt*: Objekt in neuer Version vorhanden, nicht in alter
  - *Gelöscht*: Objekt in alter Version vorhanden, nicht in neuer
  - *Verschoben*: Kilometrierung geändert (basierend auf coord\_value)
4. **Visualisierung:** Die UI stellt Änderungen durch farbkodierte Overlays dar (grün: hinzugefügt, rot: gelöscht, gelb: verschoben) mit tabellarischer Zusammenfassung und Jump-to-Detection-Funktionalität.

Die Validierung anhand zweier realer Planversionen mit 41 manuell verifizierten Änderungen bestätigt die Funktionsfähigkeit des Ansatzes. Der strukturbasierte Vergleich ist robuster gegenüber visuellen Variationen (Scanqualität, Kompressionsartefakte) als pixelbasierte Differenzbildung und ermöglicht die Kategorisierung auf semantischer Ebene.

### 8.7.6 Zusammenfassung

Alle Teilstudien konnten durch die entwickelte Lösung positiv beantwortet werden. Die Hauptforschungsfrage nach der Automatisierbarkeit der Datenextraktion aus technischen Zeichnungen wird durch die erreichte End-to-End-Genauigkeit von 97.22% (vgl. Tabelle 7.7) und Zeitersparnis von 89,7% (vgl. Tabelle 7.17) eindeutig bestätigt. Die Kombination aus Deep Learning (YOLOv8-OBB), hybrider OCR-Strategie (Multi-Engine-Kaskade) und regelbasiertem Linking stellt einen effektiven Ansatz dar, der auch mit begrenzten Trainingsdaten (24 Originalpläne) State-of-the-Art-Ergebnisse erzielt.

## 8.8 Abschließende Bewertung

Diese Masterarbeit demonstriert die Machbarkeit einer automatisierten Pipeline zur Extraktion strukturierter Daten aus technischen Gleisplänen mit State-of-the-Art-Leistung. Der entwickelte Prototyp erreicht eine Objekterkennungsgenauigkeit von mAP@0.5: 98.4% (Validierungssatz,

vgl. Tabelle 7.4) und eine End-to-End-Genauigkeit von **97.22%** (Testsatz, vgl. Tabelle 7.7), womit die Zielanforderung von 85% (NFA-003) um 12.22 Prozentpunkte übertroffen wird.

#### Zentrale Beiträge der Arbeit:

- **Technisch:** Erfolgreiche Kombination von OBB-basierter Objekterkennung (YOLOv8-OBB) mit orientierungsadaptiver Multi-Engine-OCR (PaddleOCR, Tesseract, EasyOCR) und hybrider Symbol-Text-Linking (Heuristik + adaptives Lernen). Die Rotationsinvarianz wurde durch synthetische Augmentation (10 Winkelschritte) erreicht, wobei rotierte Objekte ( $|\theta| > 30^\circ$ ) sogar höhere Konfidenz zeigen als horizontal ausgerichtete (0.946 vs. 0.890).
- **Methodisch:** Demonstration, dass synthetische Rotationsaugmentation begrenzte Trainingsdaten (24 Originalpläne → 240 augmentierte Varianten) effektiv kompensieren kann. Die erreichte Genauigkeit ist vergleichbar mit Systemen, die auf wesentlich größeren Datensätzen trainiert wurden.
- **Praktisch:** Funktionsfähiger Prototyp mit PyQt5-basierter Benutzeroberfläche für realistische Workflows. Die Zeitersparnis von 89,7% (102 min → 10,6 min pro Plan, vgl. Tabelle 7.17) und die Human-in-the-Loop-Validierung ermöglichen effizienten Produktiveinsatz.
- **Wissenschaftlich:** Systematische Evaluation auf drei Ebenen (YOLO-Detection, OCR/Linking-Komponenten, End-to-End-System) mit transparenter Fehleranalyse. Die Identifikation spezifischer Fehlerursachen (OCR: 1.83%, Linking: 0.88%, Detection: 0.07%, vgl. Tabelle 7.12) ermöglicht gezielte Verbesserungen.

#### Limitationen und Herausforderungen:

Die kritische Reflexion offenbart mehrere Einschränkungen:

- **Layout-Spezialisierung:** Das System ist auf Siemens Trainguard MT ZUB Layouts optimiert und benötigt Nachtraining für andere Bahnbetreiber oder Sicherungssysteme.
- **Auflösungsabhängigkeit:** Die strikte 500 DPI Anforderung begrenzt die Flexibilität; niedrigere Auflösungen führen zu drastisch reduzierter Genauigkeit.
- **Verbleibende Fehlerrate:** 2.78% der Objekte erfordern manuelle Korrektur, was vollautomatischen Betrieb verhindert. Dies ist für sicherheitskritische Anwendungen jedoch akzeptabel – der Prototyp ist als Assistenzsystem konzipiert.
- **Tile-Boundary-Probleme:** Objekte an Kachelgrenzen verursachen gelegentlich Duplikate oder fehlerhafte Linking-Entscheidungen.
- **Fehlendes semantisches Verständnis:** Keine Prüfung von Plausibilität (Kilometrierung-Monotonie, Duplikate, technische Abhängigkeiten).

#### Verbesserungsperspektiven:

Die identifizierten Limitationen sind adressierbar durch:

- **Kurzfristig (0-6 Monate):** OCR-Optimierungen, erweiterte Linking-Heuristiken, semantische Validierung → Ziel: 99.2-99.5% E2E-Accuracy
- **Mittelfristig (6-12 Monate):** Neuronales Linking-Modell (GNN/Attention), GPU-Beschleunigung, Active Learning → Ziel: 99.5% Accuracy, < 3 min Verarbeitungszeit
- **Langfristig (12+ Monate):** Multimodales End-to-End-Lernen, Transfer Learning, Integration in digitale Zwillinge → Ziel: 99.8% Accuracy, vollautomatische Workflows

### **Übertragbarkeit:**

Die entwickelten Konzepte sind zu 60-70% auf andere technische Zeichnungsdomänen (P&ID, Elektropläne, Hydraulikschaltpläne) übertragbar. Insbesondere die OBB-basierte Architektur, Multi-Engine-OCR und modulare Pipeline-Struktur sind domänenübergreifend anwendbar.

### **Wissenschaftlicher Erkenntnisgewinn:**

Diese Arbeit validiert den Ansatz, domänenspezifische Computer-Vision-Lösungen mit modernen Deep-Learning-Methoden und gezielten Heuristiken zu kombinieren. Dieser *Hybrid-Ansatz* stellt einen pragmatischen Mittelweg dar zwischen:

- **Rein datengetriebenen End-to-End-Modellen**, die massive Datenmengen (Tausende annotierte Beispiele) und Rechenressourcen (GPU-Cluster) erfordern, aber höchste Genauigkeit versprechen
- **Rein regelbasierten Systemen**, die mit wenig Daten auskommen, aber schwer wartbar, nicht robust und limitiert in der Generalisierung sind

Die erfolgreiche Evaluation auf realen Siemens Mobility Daten (vgl. Kapitel 7) belegt, dass der entwickelte Prototyp das Potenzial hat, manuelle Prozesse signifikant zu beschleunigen (89,7% Zeitersparnis, vgl. Tabelle 7.17) und damit einen konkreten Beitrag zur Digitalisierung der Eisenbahninfrastruktur zu leisten. Die hohe Genauigkeit (97.22%, vgl. Tabelle 7.7), kombiniert mit transparenten Validierungsmechanismen und Human-in-the-Loop-Integration, positioniert das System als praktikable Lösung für den produktiven Einsatz in sicherheitskritischen Umgebungen.

### **Ausblick:**

Die nächsten Schritte umfassen die Pilotierung in realen Projekten (Q2 2026), systematische Datensammlung aus dem Produktiveinsatz und iterative Modellverbesserung. Langfristig könnte dieses System die Grundlage für umfassendere digitale Zwillinge der Bahninfrastruktur bilden, die Planungsdaten, Simulationsmodelle und operative Sensordaten integrieren.

Die vorliegende Arbeit demonstriert, dass KI-gestützte Dokumentenverarbeitung für hochspezialisierte technische Domänen mit überschaubarem Datenaufwand und Standard-Hardware realisierbar ist – ein ermutigendes Ergebnis für die Digitalisierung weiterer ingenieurtechnischer Prozesse.

# Literaturverzeichnis

- [1] Qi Sun u. a. „Symbol Recognition Method for Railway Catenary Layout Drawings Based on Deep Learning“. In: *Symmetry* 17.5 (2025). ISSN: 2073-8994. DOI: 10.3390/sym17050674. URL: <https://www.mdpi.com/2073-8994/17/5/674>.
- [2] Ivan Ristic. „Automated Development of Railway Signalling Control Tables: A Case Study from Serbia“. In: *Mechatronics and Intelligent Transportation Systems* (2023). DOI: <https://doi.org/10.56578/mits020404>.
- [3] Birgit Vogel-Heuser, Thomas Bauernhansl und Michael ten Hompel. *Handbuch Industrie 4.0 Bd.2: Automatisierung*. Berlin, Heidelberg: Springer Vieweg, 2017. ISBN: 978-3-662-53249-6.
- [4] Yuan Wang, Xiaopeng Li und Yu Zhang. „An automation solution to convert CAD engineering drawings into railroad station models“. In: *Computer-Aided Civil and Infrastructure Engineering* 39.5 (2024), S. 679–691. DOI: <https://doi.org/10.1111/mice.13091>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.13091>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.13091>.
- [5] Jörn Pachl. *Systemtechnik des Schienenverkehrs: Bahnbetrieb planen, steuern und sichern*. Jan. 2021. ISBN: 978-3-658-31164-3. DOI: 10.1007/978-3-658-31165-0.
- [6] Deutsche Bahn AG. *Richtlinie 819 - Gleispläne: Betriebliche und technische Anforderungen, Gestaltung*. Frankfurt am Main, 2008.
- [7] Bjørnar Luteberget und Christian Johansen. „Efficient verification of railway infrastructure designs against standard regulations“. In: *Form. Methods Syst. Des.* 52.1 (Feb. 2018), S. 1–32. ISSN: 0925-9856. DOI: 10.1007/s10703-017-0281-z. URL: <https://doi.org/10.1007/s10703-017-0281-z>.
- [8] Y. Lecun u. a. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.
- [9] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. „ImageNet classification with deep convolutional neural networks“. In: *Commun. ACM* 60.6 (Mai 2017), S. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: <https://doi.org/10.1145/3065386>.
- [10] I. Zeid. *Mastering CAD/CAM*. McGraw-Hill Higher Education, 2005. ISBN: 9780072976816. URL: <https://books.google.de/books?id=wJ91QgAACAAJ>.
- [11] Deutsches Institut für Normung. *DIN ISO 128-1: Technische Produktdokumentation – Allgemeine Grundlagen der Darstellung*. Berlin, 2020.
- [12] Tsung-Yi Lin u. a. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: <https://arxiv.org/abs/1405.0312>.
- [13] Mark Everingham u. a. „The Pascal Visual Object Classes (VOC) challenge“. In: *International Journal of Computer Vision* 88 (Juni 2010), S. 303–338. DOI: 10.1007/s11263-009-0275-4.

- [14] Sheraz Ahmed u. a. „Automatic Room Detection and Room Labeling from Architectural Floor Plans“. In: März 2012, S. 339–343. ISBN: 9780769546612. DOI: 10.1109/DAS.2012.22.
- [15] Carlos Moreno-García, Eyad Elyan und Chrisina Jayne. „New trends on digitisation of complex engineering drawings“. In: *Neural Computing and Applications* 31 (Juni 2019). DOI: 10.1007/s00521-018-3583-1.
- [16] Rohit Rahul u. a. „Automatic Information Extraction from Piping and Instrumentation Diagrams“. In: *ArXiv* abs/1901.11383 (2019). URL: <https://api.semanticscholar.org/CorpusID:59523595>.
- [17] Laura Jamieson, Carlos Moreno-García und Eyad Elyan. „A review of deep learning methods for digitisation of complex documents and engineering diagrams“. In: *Artificial Intelligence Review* 57 (Mai 2024). DOI: 10.1007/s10462-024-10779-2.
- [18] Yann LeCun, Y. Bengio und Geoffrey Hinton. „Deep Learning“. In: *Nature* 521 (Mai 2015), S. 436–44. DOI: 10.1038/nature14539.
- [19] Zhengxia Zou u. a. „Object Detection in 20 Years: A Survey“. In: *Proceedings of the IEEE* 111.3 (2023), S. 257–276. DOI: 10.1109/JPROC.2023.3238524.
- [20] Shaoqing Ren u. a. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“. In: *Advances in Neural Information Processing Systems*. Hrsg. von C. Cortes u. a. Bd. 28. Curran Associates, Inc., 2015. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf).
- [21] Joseph Redmon u. a. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: <https://arxiv.org/abs/1506.02640>.
- [22] Glenn Jocher, Ayush Chaurasia und Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: <https://github.com/ultralytics/ultralytics>.
- [23] Tsung-Yi Lin u. a. „Feature Pyramid Networks for Object Detection“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2017.
- [24] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: <https://arxiv.org/abs/2408.15857>.
- [25] Alexey Bochkovskiy, Chien-Yao Wang und Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: <https://arxiv.org/abs/2004.10934>.
- [26] Jian Ding u. a. „Learning Rot Transformer for Oriented Object Detection in Aerial Images“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, S. 2844–2853. DOI: 10.1109/CVPR.2019.00296.
- [27] Jianqi Ma u. a. „Arbitrary-Oriented Scene Text Detection via Rotation Proposals“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, S. 3892–3900. DOI: 10.1109/CVPR.2018.00410. URL: <https://doi.org/10.1109/CVPR.2018.00410>.

- [28] Tsung-Yi Lin u. a. „Focal Loss for Dense Object Detection“. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, S. 2999–3007. DOI: 10.1109/ICCV.2017.324. URL: <https://doi.org/10.1109/ICCV.2017.324>.
- [29] Xingxing Xie u. a. „Oriented R-CNN for Object Detection“. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, S. 3500–3509. DOI: 10.1109/ICCV48922.2021.00350. URL: <https://doi.org/10.1109/ICCV48922.2021.00350>.
- [30] Xinyu Zhou u. a. „EAST: An Efficient and Accurate Scene Text Detector“. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 2642–2651. DOI: 10.1109/CVPR.2017.283. URL: <https://doi.org/10.1109/CVPR.2017.283>.
- [31] Xue Yang u. a. *R3Det: Refined Single-Stage Detector with Feature Refinement for Rotating Object*. 2021. DOI: 10.48550/arXiv.1908.05612. arXiv: 1908.05612 [cs.CV]. URL: <https://arxiv.org/abs/1908.05612>.
- [32] Zifei Zhao und Shengyang Li. „ABFL: Angular Boundary Discontinuity Free Loss for Arbitrary Oriented Object Detection in Aerial Images“. In: *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024), S. 1–11. DOI: 10.1109/TGRS.2024.3368630.
- [33] Chien-Yao Wang u. a. „CSPNet: A New Backbone That Can Enhance Learning Capability of CNN“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, S. 1571–1580. DOI: 10.1109/CVPRW50498.2020.00203. URL: <https://doi.org/10.1109/CVPRW50498.2020.00203>.
- [34] Shu Liu u. a. „Path Aggregation Network for Instance Segmentation“. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, S. 8759–8768. DOI: 10.1109/CVPR.2018.00913. URL: <https://doi.org/10.1109/CVPR.2018.00913>.
- [35] Zhi Tian u. a. „FCOS: Fully Convolutional One-Stage Object Detection“. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, S. 9626–9635. DOI: 10.1109/ICCV.2019.00972.
- [36] Zheng Ge u. a. *YOLOX: Exceeding YOLO Series in 2021*. 2021. DOI: 10.48550/arXiv.2107.08430. arXiv: 2107.08430 [cs.CV]. URL: <https://arxiv.org/abs/2107.08430>.
- [37] Xiang Li u. a. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*. 2020. arXiv: 2006.04388 [cs.CV]. URL: <https://arxiv.org/abs/2006.04388>.
- [38] Karl Tombre u. a. „Text/Graphics Separation Revisited“. In: *Document Analysis Systems* V. Hrsg. von Daniel Lopresti, Jianying Hu und Ramanujan Kashi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 200–211. ISBN: 978-3-540-45869-2.
- [39] Minghui Liao u. a. *Real-time Scene Text Detection with Differentiable Binarization*. 2019. arXiv: 1911.08947 [cs.CV]. URL: <https://arxiv.org/abs/1911.08947>.
- [40] Baoguang Shi, Xiang Bai und Cong Yao. *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*. 2017. arXiv: 1507.05717 [cs.CV]. URL: <https://arxiv.org/abs/1507.05717>.

- [41] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Comput.* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [42] Alex Graves u. a. „Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks‘“. In: Bd. 2006. Jan. 2006, S. 369–376. DOI: 10.1145/1143844.1143891.
- [43] Ashish Vaswani u. a. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762>.
- [44] Minghao Li u. a. *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. 2022. arXiv: 2109.10282 [cs.CL]. URL: <https://arxiv.org/abs/2109.10282>.
- [45] Alexey Dosovitskiy u. a. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: <https://arxiv.org/abs/2010.11929>.
- [46] Yuning Du u. a. *PP-OCR: A Practical Ultra Lightweight OCR System*. 2020. arXiv: 2009.09941 [cs.CV]. URL: <https://arxiv.org/abs/2009.09941>.
- [47] R. Smith. „An Overview of the Tesseract OCR Engine“. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Bd. 2. 2007, S. 629–633. DOI: 10.1109/ICDAR.2007.4376991.
- [48] Youngmin Baek u. a. *Character Region Awareness for Text Detection*. 2019. arXiv: 1904.01941 [cs.CV]. URL: <https://arxiv.org/abs/1904.01941>.
- [49] George Nagy, Thomas Nartker und Stephen Rice. „Optical character recognition: an illustrated guide to the frontier“. In: *Proceedings of SPIE - The International Society for Optical Engineering* 3967 (Dez. 1999), S. 58–69. DOI: 10.1117/12.373511.
- [50] Andrew Morris, Viktoria Maier und Phil Green. „From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition“. In: Okt. 2004. DOI: 10.21437/Interspeech.2004-668.
- [51] Tobias Schlagenhauf, Markus Netzer und Jan Hillinger. *Text Detection on Technical Drawings for the Digitization of Brown-field Processes*. 2022. arXiv: 2205.02659 [cs.CV]. URL: <https://arxiv.org/abs/2205.02659>.
- [52] Max Jaderberg u. a. *Spatial Transformer Networks*. 2016. arXiv: 1506.02025 [cs.CV]. URL: <https://arxiv.org/abs/1506.02025>.
- [53] Rafael Gonzalez und Zahraa Faisal. *Digital Image Processing Second Edition*. Juni 2019.
- [54] Marcelo Bertalmio, Guillermo Sapiro und C. Ballester. „Image Inpainting“. In: *Proceedings of SIGGRAPH* (Jan. 2002). DOI: 10.1145/344779.344972.
- [55] Vladimir I. Levenshtein. „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet Physics Doklady* 10.8 (1966), S. 707–710.
- [56] Shangbang Long, Xin He und Cong Yao. *Scene Text Detection and Recognition: The Deep Learning Era*. 2020. arXiv: 1811.04256 [cs.CV]. URL: <https://arxiv.org/abs/1811.04256>.

- [57] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2. Aufl. Texts in Computer Science. Also available as eBook: ISBN 978-3-030-34372-9. Springer Cham, 2022, S. xxii+925. ISBN: 978-3-030-34371-2. DOI: 10 . 1007 / 978 - 3 - 030 - 34372 - 9. URL: <https://doi.org/10.1007/978-3-030-34372-9>.
- [58] Max Wertheimer. „Untersuchungen zur Lehre von der Gestalt. II“. In: *Psychologische Forschung* 4 (), S. 301–350. URL: <https://api.semanticscholar.org/CorpusID:143510308>.
- [59] Koichi Kise, Akinori Sato und Motoi Iwata. „Segmentation of Page Images Using the Area Voronoi Diagram“. In: *Computer Vision and Image Understanding* 70.3 (1998), S. 370–382. ISSN: 1077-3142. DOI: <https://doi.org/10.1006/cviu.1998.0684>. URL: <https://www.sciencedirect.com/science/article/pii/S1077314298906841>.
- [60] Song Mao, Azriel Rosenfeld und Tapas Kanungo. „Document structure analysis algorithms: a literature survey“. In: *Document Recognition and Retrieval X*. Hrsg. von Tapas Kanungo u. a. Bd. 5010. International Society for Optics und Photonics. SPIE, 2003, S. 197–207. DOI: 10.1117/12.476326. URL: <https://doi.org/10.1117/12.476326>.
- [61] Dietmar Gross u. a. *Technische Mechanik 1: Statik*. 14. Aufl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. ISBN: 978-3-662-59156-7. DOI: 10.1007/978-3-662-59157-4.
- [62] Bruno Siciliano u. a. *Robotics: Modelling, Planning and Control*. London: Springer London, 2009. ISBN: 978-1-84628-642-1. DOI: 10.1007/978-1-84628-642-1.
- [63] Ulrich Kurz und Herbert Wittel. *Böttcher/Forberg Technisches Zeichnen: Grundlagen, Normung, Übungen und Projektaufgaben*. 26. Aufl. Wiesbaden: Springer Vieweg, 2014. ISBN: 978-3-8348-2232-1. DOI: 10.1007/978-3-8348-2232-1.
- [64] DIN EN ISO 9001:2015 Qualitätsmanagementsysteme – Anforderungen. ISO 9001:2015(E). Deutsches Institut für Normung e. V. (DIN), 2015.
- [65] John Stark. *Product Lifecycle Management (Volume 1): 21st Century Paradigm for Product Realisation*. 5. Aufl. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-98578-3. DOI: 10.1007/978-3-030-98578-3.
- [66] Harold W. Kuhn. „The Hungarian Method for the Assignment Problem“. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), S. 83–97. DOI: 10.1002/nav.3800020109.
- [67] James Munkres. „Algorithms for the Assignment and Transportation Problems“. In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957), S. 32–38. DOI: 10.1137/0105003.
- [68] Rainer Burkard, Mauro Dell'Amico und Silvano Martello. *Assignment Problems: Revised Reprint*. Society for Industrial und Applied Mathematics, 2012. ISBN: 978-1-611972-22-1. DOI: 10.1137/1.9781611972238.
- [69] Edsger W. Dijkstra. „On the Role of Scientific Thought“. In: *Selected Writings on Computing: A personal Perspective*. New York, NY: Springer New York, 1982, S. 60–66. ISBN: 978-1-4612-5695-3. DOI: 10 . 1007 / 978 - 1 - 4612 - 5695 - 3 \_ 12. URL: [https://doi.org/10.1007/978-1-4612-5695-3\\_12](https://doi.org/10.1007/978-1-4612-5695-3_12).

- [70] Jami J. Shah und Martti Mäntylä. „Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications“. In: 1995. URL: <https://api.semanticscholar.org/CorpusID:264685016>.
- [71] International Electrotechnical Commission. *IEC 61131-3: Programmable controllers – Part 3: Programming languages*. Geneva, 2013.
- [72] „The Finite Element Method: Its Basis and Fundamentals“. In: *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*. Hrsg. von O.C. Zienkiewicz, R.L. Taylor und J.Z. Zhu. Seventh Edition. Oxford: Butterworth-Heinemann, 2013, S. i. ISBN: 978-1-85617-633-0. DOI: <https://doi.org/10.1016/B978-1-85617-633-0.00019-8>. URL: <https://www.sciencedirect.com/science/article/pii/B9781856176330000198>.
- [73] Olly Gotel und A. C. W. Finkelstein. „An analysis of the requirements traceability problem“. In: *Proceedings of IEEE International Conference on Requirements Engineering* (1994), S. 94–101. URL: <https://api.semanticscholar.org/CorpusID:5870868>.
- [74] European Parliament and Council. *Regulation (EC) No 178/2002: General principles and requirements of food law*. Official Journal of the European Communities, L 31/1. 2002.
- [75] U.S. Food and Drug Administration. *21 CFR Part 11: Electronic Records; Electronic Signatures*. Code of Federal Regulations, Title 21. 1997.
- [76] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Jan. 2010. ISBN: 978-3-642-12577-5. DOI: 10.1007/978-3-642-12578-2.
- [77] J. M. Juran und Frank M. Gryna, Hrsg. *Juran's Quality Control Handbook*. 4. Aufl. New York: McGraw-Hill, 1988. ISBN: 978-0-07-033176-1.
- [78] Chuan Guo u. a. „On Calibration of Modern Neural Networks“. In: *ArXiv abs/1706.04599* (2017). URL: <https://api.semanticscholar.org/CorpusID:28671436>.
- [79] Robert Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Shelter Island, NY: Manning Publications, 2021. ISBN: 978-1-61729-674-1.
- [80] Navaneeth Bodla u. a. „Improving Object Detection With One Line of Code“. In: (Apr. 2017). DOI: 10.48550/arXiv.1704.04503.
- [81] Edouard Belval. *pdf2image: A python wrapper for pdftoppm and pdftocairo*. <https://github.com/Belval/pdf2image>. GitHub repository. Accessed: 2024-05-20. 2024.
- [82] Poppler Development Team. *Poppler PDF rendering library*. <https://poppler.freedesktop.org/>. Accessed: 2024-05-20. 2024.
- [83] Artifex Software Inc. *PyMuPDF Documentation – License*. AGPL-3.0 License, accessed 2024. 2024. URL: <https://pymupdf.readthedocs.io/en/latest/about.html>.
- [84] Yusuke Shinyama. *PDFMiner: Python PDF Parser*. <https://github.com/pdfminer/pdfminer.six>. Version 20221105. Accessed: 2024-05-20. 2022.
- [85] Ultralytics. *YOLOv8 Oriented Bounding Boxes (OBB)*. <https://docs.ultralytics.com/tasks/obb/>. Accessed: 2024-05-20. 2024.
- [86] Chien-Yao Wang, Alexey Bochkovskiy und Hong-Yuan Mark Liao. „YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors“. In: 2023

- IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2023, S. 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.
- [87] Chengqi Lyu u. a. *RTMDet: An Empirical Study of Designing Real-Time Object Detectors*. 2022. arXiv: 2212.07784 [cs.CV]. URL: <https://arxiv.org/abs/2212.07784>.
- [88] Roberto Brunelli. „Template Matching Techniques in Computer Vision“. In: (Sep. 2008).
- [89] Nicolas Carion u. a. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV]. URL: <https://arxiv.org/abs/2005.12872>.
- [90] PaddlePaddle Team. *Paddleocr: Multilingual, Lightweight, and Practical Optical Character Recognition*. <https://www.paddleocr.ai/latest/>. Accessed: 2025-11-21. 2023.
- [91] Nobuyuki Otsu. „A Threshold Selection Method from Gray-Level Histograms“. In: *IEEE Trans. Syst. Man Cybern.* 9.1 (1979), S. 62–66. DOI: 10.1109/tsmc.1979.4310076.
- [92] Yupan Huang u. a. *LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking*. 2022. arXiv: 2204.08387 [cs.CL]. URL: <https://arxiv.org/abs/2204.08387>.
- [93] Geewook Kim u. a. *OCR-free Document Understanding Transformer*. 2022. arXiv: 2111.15664 [cs.LG]. URL: <https://arxiv.org/abs/2111.15664>.
- [94] openpyxl contributors. *openpyxl: A Python library to read/write Excel 2010 xlsx/xlsm files*. 2024. URL: <https://openpyxl.readthedocs.io>.
- [95] John McNamara. *XlsxWriter Documentation*. Python module for creating Excel XLSX files. 2024. URL: [https://xlsxwriter.readthedocs.io/](https://xlsxwriter.readthedocs.io).
- [96] VDI 1000:2017-06 *VDI-Richtlinienarbeit – Grundsätze und Anleitungen*. VDI-Richtlinie. Verein Deutscher Ingenieure (VDI), 2017.
- [97] Riverbank Computing. *PyQt5 Reference Guide*. 2024. URL: <https://www.riverbankcomputing.com/static/Docs/PyQt5/>.
- [98] Streamlit Inc. *Streamlit Security Best Practices and Deployment*. <https://docs.streamlit.io/deploy>. Accessed: 2024-05-20. 2024.
- [99] Pallets Projects. *Flask Documentation*. Python micro web framework. 2024. URL: <https://flask.palletsprojects.com/>.
- [100] OpenJS Foundation. *Electron Documentation*. Framework for building cross-platform desktop apps with JavaScript. 2024. URL: <https://www.electronjs.org/docs/>.
- [101] Python Software Foundation. *Tkinter – Python interface to Tcl/Tk*. Python's standard GUI library. 2024. URL: <https://docs.python.org/3/library/tkinter.html>.
- [102] PostgreSQL 15 Documentation. The PostgreSQL Global Development Group. 2024. URL: <https://www.postgresql.org/docs/15/>.
- [103] SQLite Development Team. *Appropriate Uses For SQLite*. <https://sqlite.org/wentouse.html>. Accessed: 2024-05-20. 2024.
- [104] MongoDB Inc. *MongoDB Documentation*. Document-oriented NoSQL database. 2024. URL: <https://www.mongodb.com/docs/>.
- [105] Intel Corporation. *CVAT: Computer Vision Annotation Tool*. 2024. URL: <https://github.com/opencv/cvat>.

- [106] International Organization for Standardization. *ISO 8601:2019 – Date and time – Representations for information interchange*. International standard for date and time formats. 2019. URL: <https://www.iso.org/standard/70907.html>.
- [107] Eun-Seop Yu u. a. „Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network“. In: *Energies* 12.23 (2019), S. 4425. DOI: 10.3390/en12234425.
- [108] Eyad Elyan, Laura Jamieson und Adamu Ali-Gombe. „Deep learning for symbols detection and classification in engineering drawings“. In: *Neural Networks* 129 (2020), S. 91–102. DOI: 10.1016/j.neunet.2020.05.025.