Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
**Maschinenbau und
Mechatronik**

PRÜFUNGSVORLEISTUNG

VDKI – Projektreport

- Vernetzte Produktionssysteme SS23 -

Von

Julia Riffel, 88382	riju1022@h-ka.de
Daniel Hettich, 88397	heda1051@h-ka.de
Sebastian Gahm, 63343	gase1012@h-ka.de
Felix Theiss, 88551	thfe1014@h-ka.de

Prüfer

Prof. Dr.-Ing. Peter Offermann

Inhaltsverzeichnis

1. Eigenschaften der Daten

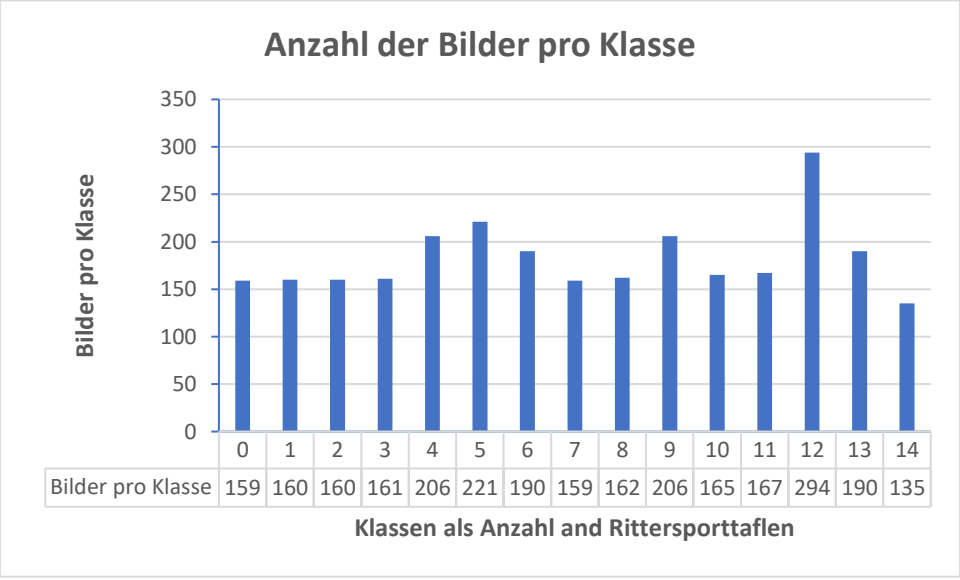
Um eine künstliche Intelligenz, kurz KI, zu trainieren und zu testen können, die darauf abzielt, die Anzahl der Rittersporttafeln auf einem Bild zu zählen, sind Bilddaten erforderlich. Diese Daten müssen entsprechend den Klassen gekennzeichnet werde. Zudem ist es notwendig Merkmale aus den Daten zu erheben. Im folgenden Abschnitt werden zunächst die Eigenschaften der verwendeten Daten erläutert und anschließend auf die Data Augmentation eingegangen.

Die Bilder weisen folgende Eigenschaften auf:

- 8 bit RGB
- 512 Pixelhöhe und 512 Pixelbreite
- Weißgrauer Hintergrund (kleine Anzahl an Bilder mit abweichender Hintergrundfarbe)
- Natürliche Belichtung
- undefinierte Position der Kamera zu Rittersporttafeln

Um die gewünschte Pixelgröße für die Bilder zu erreichen, wurde die OpenCV-Bibliothek verwendet. Hierfür wurden die Bilder mit der resize()-Funktion auf die benötigte Pixelgröße skaliert. Bei der gewählten Auflösung sind Merkmale für die Klassifikation noch ausreichend gut erkennbar und gleichzeitig wird der benötigte Speicherplatz, die Speicher- und Ladezeiten als auch die Rechenzeit reduziert. Der weißgraue Hintergrund und die natürliche Belichtung wurden auf Grund des im Rahmen der Vorlesung gezeigten Testfoto-Setup gewählt. Es wurde eine Gesamtanzahl von 15 Klassen festgelegt, in der die Unterscheidung von 0 bis 14 Rittersporttafeln enthalten ist. Durch die Begrenzung der Klassenanzahl auf 15 kann die Erstellung der Bilddaten effizienter gestaltet werden, da nur eine bestimmte Anzahl von Rittersporttafeln berücksichtigt wird. Somit müssen nicht unendlich viele Bilder mit unterschiedlichen Anzahlen von Tafeln erstellt werden. Dies ermöglicht eine zielgerichtete Erfassung der Merkmale der Rittersporttafeln für das Training und die Testung der KIs. Somit können Zeit und Ressourcen bei der Datenerstellung eingespart werden.

Die Aufteilung der Bilder pro Klasse ist in der Abbildung 1 dargestellt. Weitere Erläuterung zu den Merkmalen finden sich in Abschnitt 2 Merkmale.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Bilder pro Klasse	159	160	160	161	206	221	190	159	162	206	165	167	294	190	135

Abbildung 1 Anzahl der Bilder pro Klasse

Im Rahmen der Data Augmentation wurde versucht mit den Funktionen der OpenCV Bibliothek die Anzahl der Bilddaten zu erhöhen. Leider erwiesen sich die augmentierten Daten aufgrund des starken Kontrasts und des hohe Gradients des weißgrauen Hintergrunds als unbrauchbar. Dies liegt an der entstandenen Kantenbildung am Bildrand. Aus diesem Grund wurde die Qualität der augmentierten Daten als zu schlecht bewertet, um diese für die Merkmalerhebung und somit fürs Training und die Testung der KIs zu nutzen.

2. Merkmale

Um die klassischen KI-Verfahren zu implementieren, müssen zunächst Merkmale aus den Bilddaten erhoben werden. Anhand dieser erhobenen Merkmale erfolgten das Training und spätere Klassifizierung eines Bildes. Insgesamt wurden 15 Merkmale erhoben. Alle Merkmale sind in der Tabelle 1 dargestellt. Vor der Merkmalserhebung wird das Bild mit der programmierten Funktion apply_color_balance(), die auf OpenCV-Befehle zugreift, angepasst. Es wird ein adaptiver Histogrammausgleich auf den B-Farbkanal angewandt, mit der Auswirkung, dass die Farben einheitlicher und dunkle Bildbereich heller sind.

Tabelle 1 Merkmale und Erklärung

Nr.	Merkmal	Erklärung
1	con_num	Anzahl der Konturen unter der Verwendung der Funktion image_contours_detection(). → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
2	con_area	Summe der Flächen der Konturen unter der Verwendung der Funktion image_contours_detection(). → Je größer die Fläche, desto mehr Rittersporttafeln auf dem Bild
3	orb_num_front	Anzahl der Matches der Key-Features des ORB-Detektors von Referenzbilder zum Bild. Als Referenz Bild wurden 6 Bilder mit der Vorderseite von einzelnen Rittersporttafeln mit unterschiedlichen Farben verwendet. → Je höher die Anzahl der Matches, desto mehr Rittersporttafeln auf dem Bild
4	orb_num_back	Anzahl der Matches der Key-Features des ORB-Detektors von Referenzbilder zum Bild. Als Referenz Bild wurden 6 Bilder mit der Rückseite von einzelnen Rittersporttafeln mit unterschiedlichen Farben verwendet. → Je höher die Anzahl der Matches, desto mehr Rittersporttafeln auf dem Bild
5	cs_green	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die grünen Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
6	cs_yellow	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die gelben Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
7	cs_white	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die weißen Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild

8	cs_purple	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die lila Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
9	cs_red	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die roten Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
10	cs_blue	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die blauen Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
11	cs_brown	Anzahl der Konturen unter der Verwendung der programmierten HSV-Farbesegmentierung für die braunen Rittersporttafeln. → Je größer die Anzahl, desto mehr Rittersporttafeln sind auf dem Bild
12	mod_length	Gesamtlänge aller Konturen die bei der HSV-Farbsegmentierung erzeugt wurden. → Je länger die Summe der Konturen, desto mehr Rittersporttafeln
13	mod_length_wo_white	Gesamtlänge aller Konturen die bei der HSV-Farbsegmentierung erzeugt wurden aber ohne Beachtung der Segmentierung für weiß. → Je länger die Summe der Konturen, desto mehr Rittersporttafeln
14	mod_area	Summe aller Flächen der Konturen die bei der HSV-Farbsegmentierung erzeugt. → Je länger die Summe der Konturen, desto mehr Rittersporttafeln
15	mod_area_wo_white	Summe aller Flächen der Konturen die bei der HSV-Farbsegmentierung erzeugt wurden aber ohne Beachtung der Segmentierung für weiß. → Je länger die Summe der Konturen, desto mehr Rittersporttafeln

In der Abbildung 2 ist die Scatter-Matrix aller Bildmerkmale zu sehen. Bis auf die cs_XXX-Merkmale zeigen die meisten Bildmerkmalkombinationen eine Korrelation zu beidseitig steigenden Werten. Das bedeutet, dass je höher der Wert eines betrachteten Merkmals ist, desto größer ist der verglichene Merkmalswert. Die Korrelation zwischen den Merkmalen kann je nach Kombination der betrachteten Bildmerkmale unterschiedlich stark sein. Des Weiteren zeigen unterschiedliche Kombinat von Bildmerkmalen manchmal mehr Streuung und manchmal mit weniger. Bei vergleichender Betrachtung der cs_XXX-Merkmale untereinander, fällt auf, dass die Segmentierung der grünen Farbe gut funktioniert, während es Probleme bei der Segmentierung in anderen Farbbereichen gibt. Diese Probleme können durch die Betrachtung der Skalierung der Achsen und das vorhandene Wissen über die Bilddaten erkannt werden.

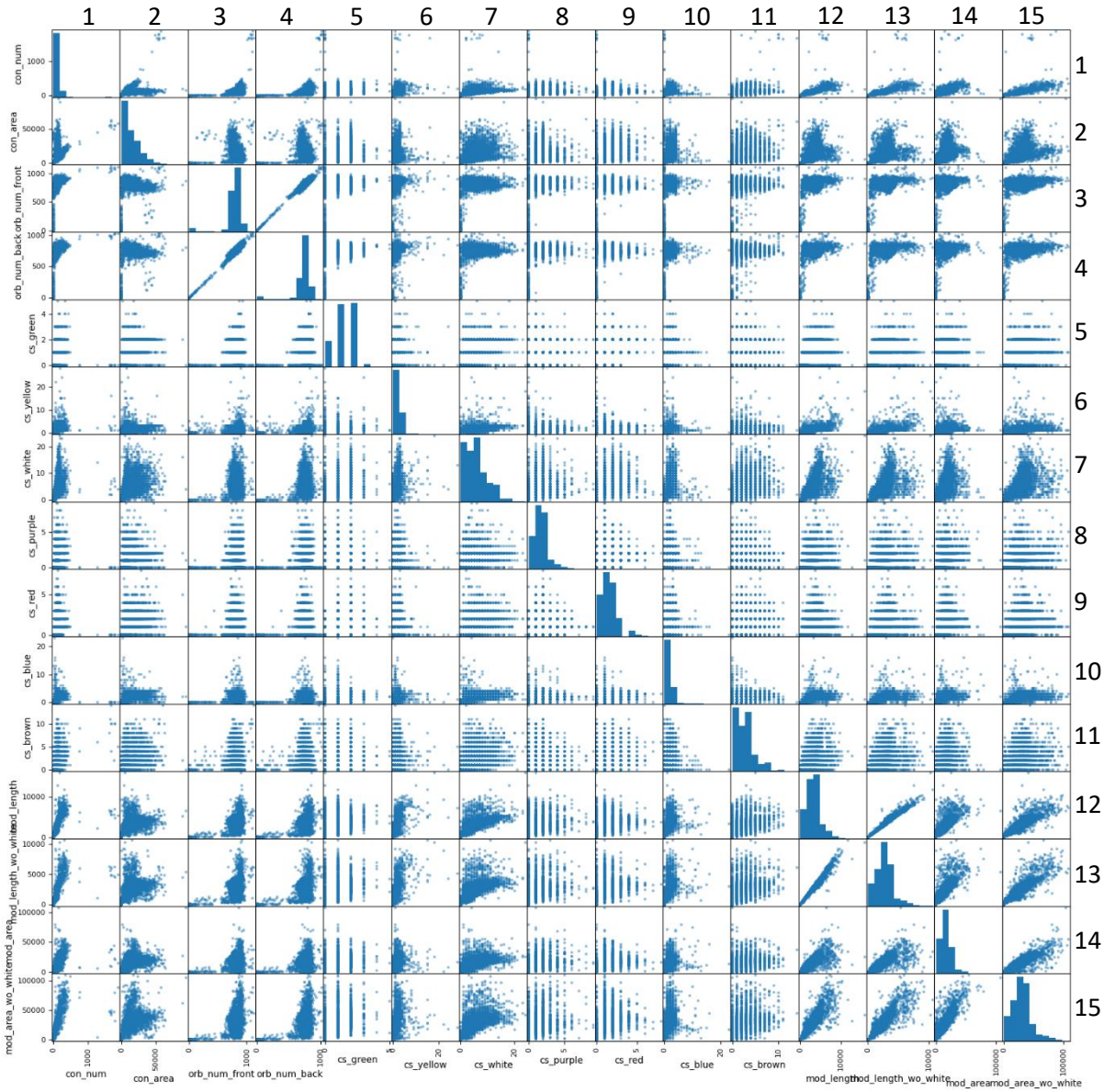


Abbildung 2 Scatter-Matrix aller Features

Im Folgenden wird auf die Besonderheiten im Quelltext für die Merkmalserhebung eingegangen.

Zum Debuggen und Visualisieren wird die Funktionen visualization_w_plt_subplot() und visualization_w_plt_subplot_gray() genutzt. Diese Funktion ermöglicht es Ergebnisse schnell und übersichtlich zu visualisieren und dabei flexibel die Zeilen- und Spaltenanzahl der Subplots anzupassen.

Für die Berechnung der Konturen nach dem Canny-Algorithmus wird die Funktion image_contours_detection() genutzt. Diese ist je nach Bedarf im Quelltext in allen Parametern anpassbar, um das gewünschte Ergebnis zu erreichen.

Um die Funktion segment_images_hsv() zur Farbsegmentierung nutzen zu können, ist es erforderlich, möglichst alle Rittersporttafeln aus dem Bild auszuschneiden. Hierfür wurde der folgende Ansatz verfolgt. Zunächst wird mit geeigneten Parametern und der Funktion image_contours_detection() eine Kontur erstellt. Anschließend werden alle Gebiete außerhalb dieser Kontur mithilfe der Funktion mask_image_outside_contour() schwarz gefärbt. Dadurch entsteht ein Bild, auf dem die Außenkontur aller Rittersporttafeln gut detektierbar ist. Um eine präzisere Segmentierung zu erreichen, wird erneut

mit passenden Parametern und der Funktion image_contours_detection() eine Kontur erstellt und wieder werden alle Gebiete außerhalb der Kontur mit der Funktion mask_image_outside_contour() schwarz gefärbt. Durch diese Technik ist es möglich, alle Rittersporttafeln vom Hintergrund auszuschneiden. Der Vorteil dieser Technik ist, dass die weißen Tafeln mit hoher Wahrscheinlichkeit aus dem ähnlichen Hintergrund ausgeschnitten werden können. Ein Nachteil dieser Methode besteht jedoch darin, dass der Ausschnitt des Bildes nicht so präzise ist. Wird die HSV-Segmentierung auf das ausgeschnittene Bild angewandt, können die voreingestellten HSV-Farbbereich entnommen und auf ein neues Bild angewandt werden. Es wird der HSV-Farbbereich genutzt, da die Farbsegmentierungsbereiche im dunklen und hellen Farbspektrum eindeutiger sind. Trotz der Verwendung des HSV-Bereiches lassen sich nicht alle Farben eindeutig isolieren, wie beispielsweise die braunen Rittersporttafeln. Deshalb werden von diesem farbsegmentierten Bild die anderen farbsegmentieren Bilder, die miterfasst wurden, abgezogen. Die resultierenden farbsegmentierten Bilder zeigen viel kleine ungewollte Farbbereiche bzw. Rauschen und werden deshalb mit den OpenCV Filterfunktionen medianBlur() und bilateralFilter() bearbeitet. Die medianBlur()-Funktion wirkt dem Rauschen entgegen und der bilateral-Filter() verringert die kleinen Farbbereiche ohne dabei die großen Farbbereiche und deren Kanten, welche die Rittersporttafeln in der entsprechenden Farbe sind, zu filtern. Bevor mit der OpenCV Funktion findContours() in den farbsegmentierten Bildern nach Konturen gesucht wird, werden die Bilder nochmals mit der threshold()-Funktion von OpenCV binarisiert, sodass die kleinen ungewollten Farbbereich noch kleiner werden. Die ermittelten Konturen werden bevor es zur Merkmalerhebung kommt, nochmal gefiltert. Beim Filtern der Konturen wird die Fläche der jeweiligen Kontur im Vergleich zur größten Flächenkontur betrachtet. Wenn die Abweichung der Flächen zu groß ist, wird die Kontur aussortiert. Somit ist es möglich eine relative konsistente Farbsegmentierung der voreingestellten Farben zu erreichen und die entsprechende Konturanzahl auszulesen.

Für detailliertere Informationen zur Programmierung und Bildbeispielen wird auf das beigefügte Jupyter Notebook verwiesen.

2.1. Nicht erhobene Merkmale

Es wurde in Erwägung gezogen, die Merkmale parallele Linien und gerade Linien aus den Konturen der Rittersporttafeln in die Merkmalerhebung einzubeziehen. Allerdings gibt es theoretische Probleme bei beiden Merkmalen. Aufgrund der perspektivischen Verzerrung ist es nicht möglich, das Merkmal parallele Linien genau und konsistent zu erfassen. Die Konturerkennung ist stark von den Parametern und der Belichtung des jeweiligen Bildes abhängig, wodurch natürliches Licht zu großen Unterschieden in der Konturerkennung führen kann.

2.2. Anforderungen zur Verbesserung der Merkmalerhebung

Im Rahmen des Projektes wurden nur wenige Anforderungen vorgegeben und es wurde den Projektgruppen freigestellt, diese selbst zu wählen. Um ein besseres Verständnis für Anforderungsmanagement im Kontext von KI zu bekommen, wurde sich gegen stark einschränkende Anforderungen entschieden, obwohl dies bedeutete, dass einige zusätzliche Herausforderungen bewältigt werden mussten.

Der Vorteil eines solchen Vorgehens mit wenigen Anforderungen ist, dass die Bilddaten von anderen Gruppen genutzt werden konnten, was zu einem breiteren Datensatz führte und eine

Arbeitszeitersparnis in der Datenerhebung erzielt werden konnte. Dennoch wurde deutlich, dass eine einheitliche Erfassung der Bilddaten und somit der Merkmale von entscheidender Bedeutung ist, um aussagekräftige und vergleichbare Ergebnisse zu erzielen. Die Erkenntnis, dass die Standardisierung der Erzeugung der Testdaten zur Merkmalsextaktion wichtig ist, wurde durch die zusätzlichen Herausforderungen, die durch die vielfältigen Herangehensweisen entstanden sind, noch deutlicher. Im Folgenden werden Anforderungen gelistet, die sowohl die Merkmalserhebung konsistenter als auch schlussendlich die KIs besser machen.

- Um die flächenbezogenen Merkmale der Rittersporttafeln konsistenter zu machen, ist es hilfreich, einen konstanten Abstand zwischen den Rittersporttafeln und der Kamera zu haben. Des Weiteren soll die Kamera senkrecht über den Rittersporttafeln positioniert sein. Dadurch wird die genannte perspektivische Verzerrung minimiert und die Fläche der Tafeln ist nur noch von den Tafeln und deren Pose abhängig.
- Eine gleichbleibende Belichtung mit diffusem Licht sollte für die Bilddaten Erzeugung genutzt werden. Ein Vorteil beim Nutzen von diffusem Licht ist, dass harte Schatten, starke Kontraste und Reflexionen reduziert werden. Die nachfolgende Bildbearbeitung und Erhebung der Merkmale sind konsistenter.
- Die Hintergrundfarbe der Bilddaten sollte sich von den Farben der Rittersporttafeln deutlich unterscheiden. Diese Anforderung erleichtert die Bildbearbeitung und die Erhebung der Merkmale, besonders unter den Aspekten der Farbsegmentierung und die Konturerkennung.

3. KI-Verfahren

Zur Erfüllung des Projektziels wurden die drei klassischen KI-Verfahren Bayesian Network, K-Nearest-Neighbour und Random Forest selbst programmiert und mit Bibliothekbefehlen von scikit-learn ausgeführt. Die vierte KI zum Zählen der Rittersporttafeln wurde mittels Convolutional Neural Network umgesetzt. Außerdem wurde ein state of the art KI, das Yolo 7 von viso.ai, auf die Testdaten angewandt. Für die Evaluierung der klassischen Verfahren werden die Kennzahlen Accuracy, Precision, Recall und F1-Score genutzt. Die erfassten Kennzahlen für die klassischen Verfahren wurden aus dem Durschnitt einer dreifachen Validierung mit zufälligen erzeugten Datenframes mit dem gleichen Trainingssplit erzeugt. Für das CNN wurden die Accuracy und Loss Kennzahl für die Trainings- und Validierungsdaten über die Epochen erstellt.

Tabelle 2 Infos zu den KI-Verfahren

KI-Verfahren	Trainingssplit	Verwendete Bildmerkmale nach Nummern
BN	20 %	1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11
KNN	20 %	alle
RF	20 %	alle
CNN	20 %	---*
Yolo 7	20 %	---*

*für das CNN und YOLO werden Bilddateien und keine Bildmerkmale verwendet

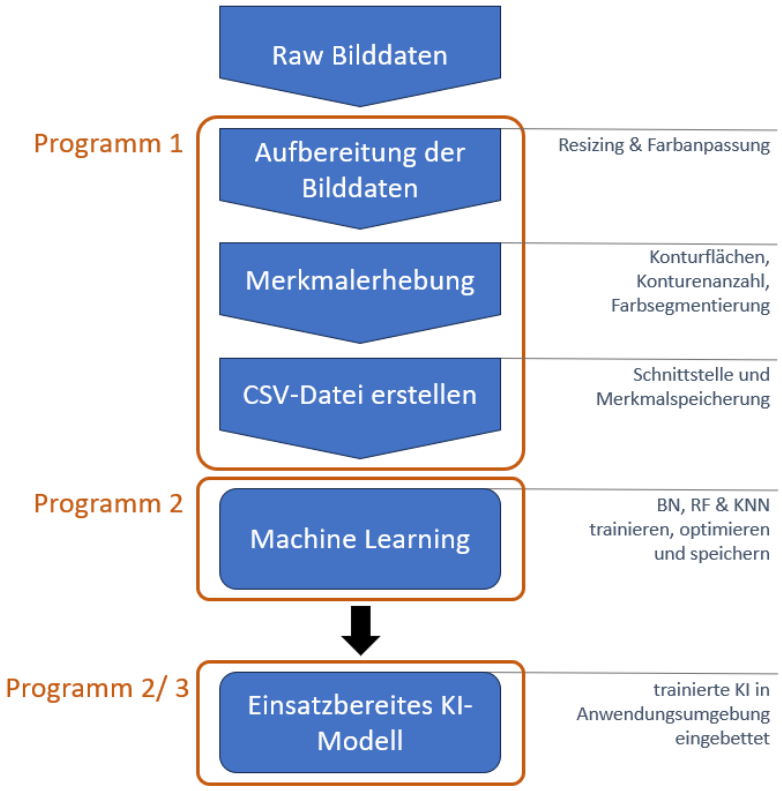


Abbildung 3 Ablaufgraph der klassischen KI-Verfahren

3.1. Bayesian Network

Tabelle 3 Evaluierung des Bayesian Network

Kennzahl	Selbstprogrammiertes BN	BN aus scikit-learn
Accuracy	52,16 %	50,76 %
Precision	50,83 %	48,71 %
Recall	52,16 %	50,76 %
F1-score	50,54 %	48,45 %

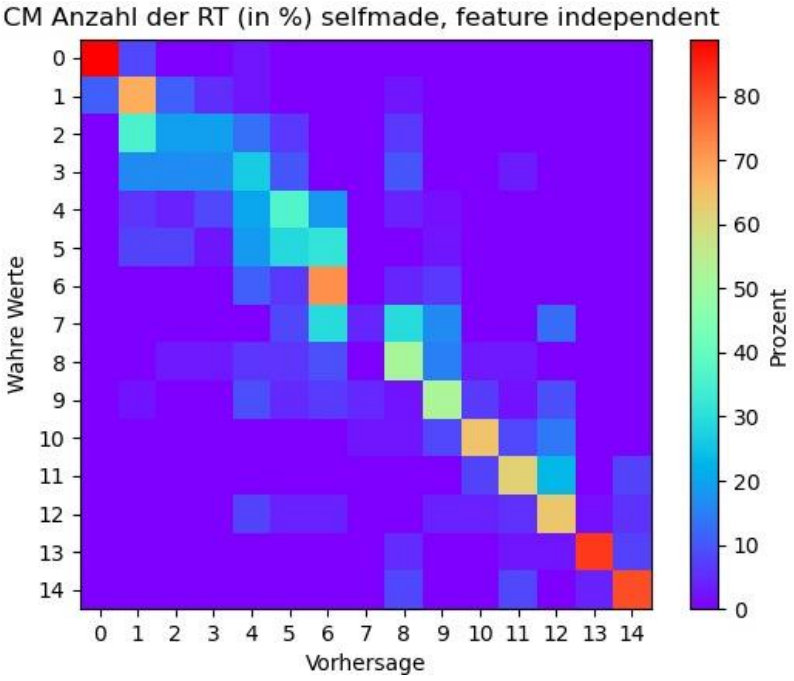


Abbildung 4 Confusion Matrix BN selbstprogrammiert

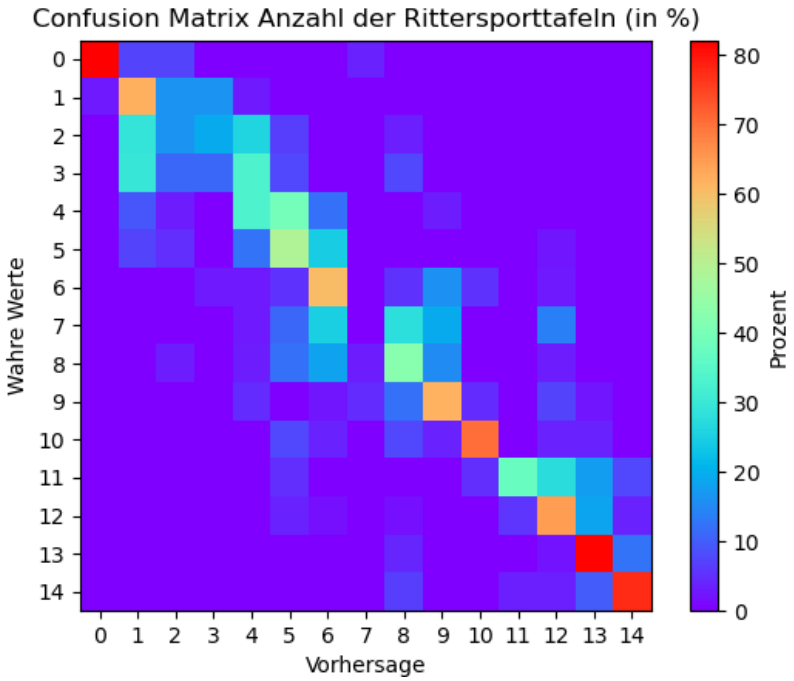


Abbildung 5 Confusion Matrix BN aus sciki-learn

3.2. K-Nearest-Neighbour

Tabelle 4 Evaluierung des K-Nearest-Neighbour

Kennzahl	Selbstprogrammiertes KNN	KNN aus scikit-learn
Accuracy	17,92 %	18,28 %
Precision	19,47 %	20,90 %
Recall	17,92 %	18,28 %
F1-score	18,42 %	18,17 %

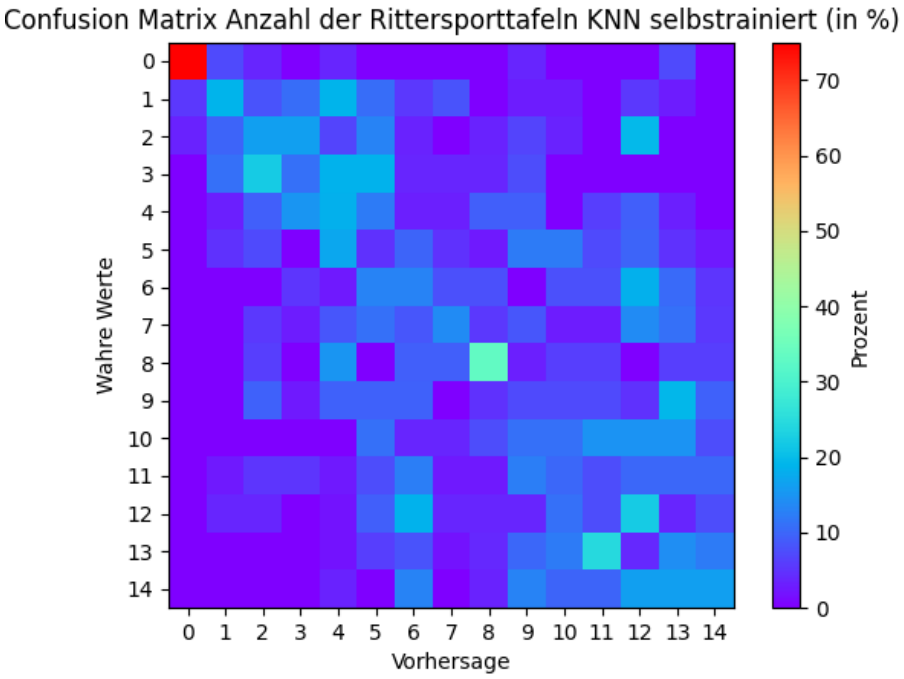


Abbildung 6 Confusion Matrix KNN selbstprogrammiert

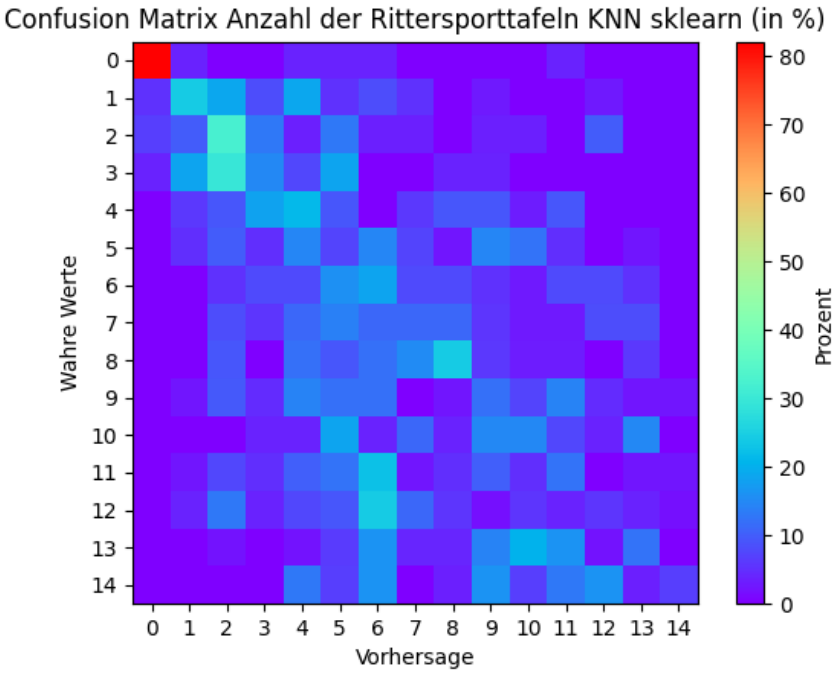


Abbildung 7 Confusion Matrix KNN aus sciki-learn

3.3. Random Forest

Tabelle 5 Evaluierung des Random Forest

Kennzahl	Selbstprogrammiertes RF	RF aus scikit-learn
Accuracy	55,52 %	64,72 %
Precision	57,31 %	65,47 %
Recall	55,52 %	64,72 %
F1-score	55,66 %	63,57 %

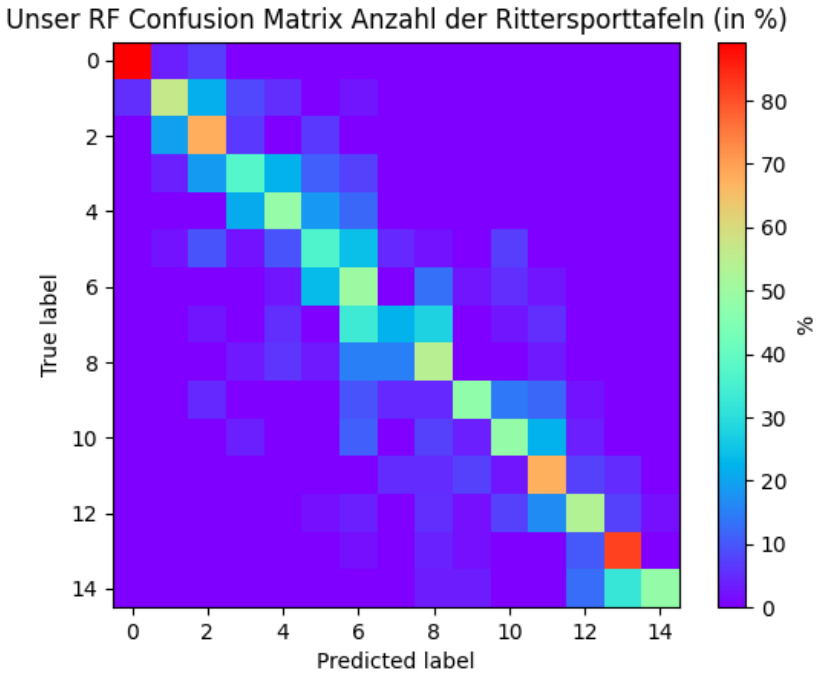


Abbildung 8 Confusion Matrix RF selbstprogrammiert

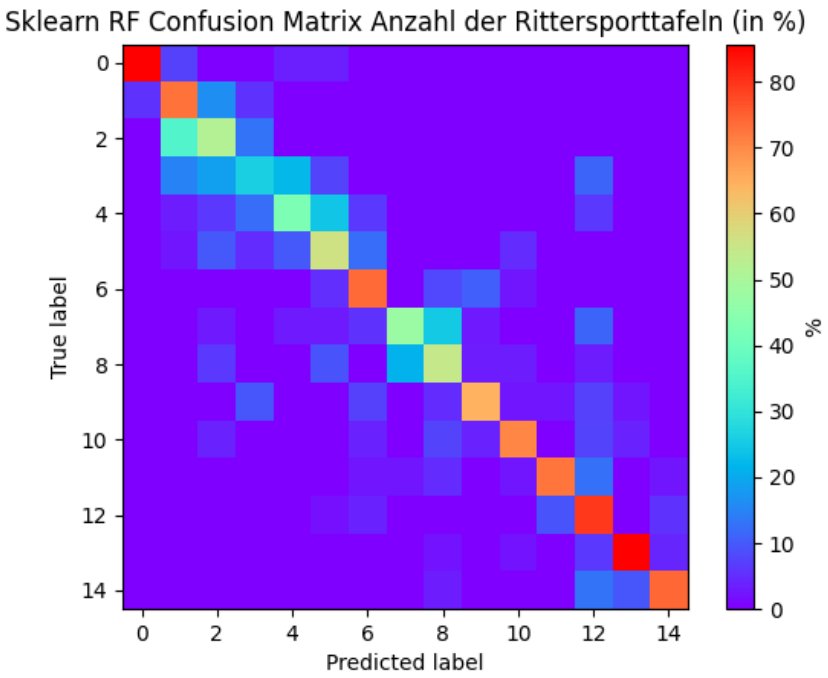


Abbildung 9 Confusion Matrix RF aus sciki-learn

3.4. Convolutional Neural Network

Tabelle 6 Evaluierung des Convolutional Neural Network

Kennzahl	Selbstprogrammiertes CNN
Accuracy	19,29 %
Precision	22,17 %
Recall	19,29 %
F1-score	17,81 %

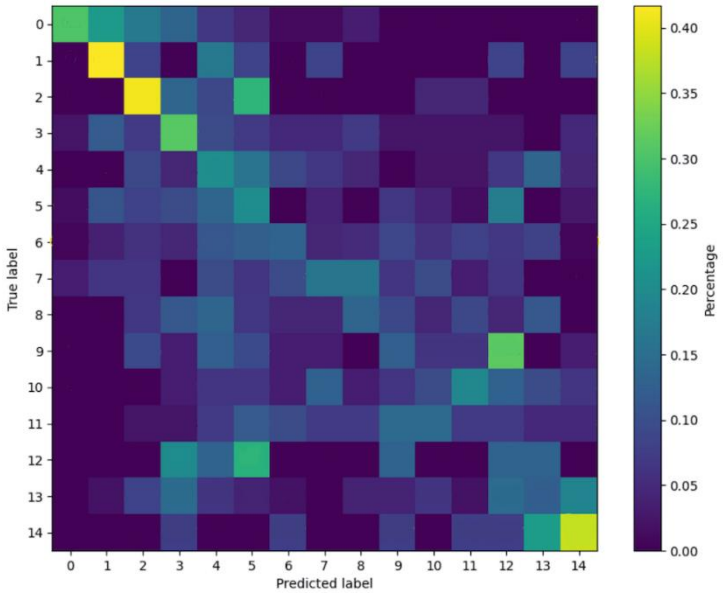


Abbildung 10 Confusion Matrix CNN selbstprogrammiert

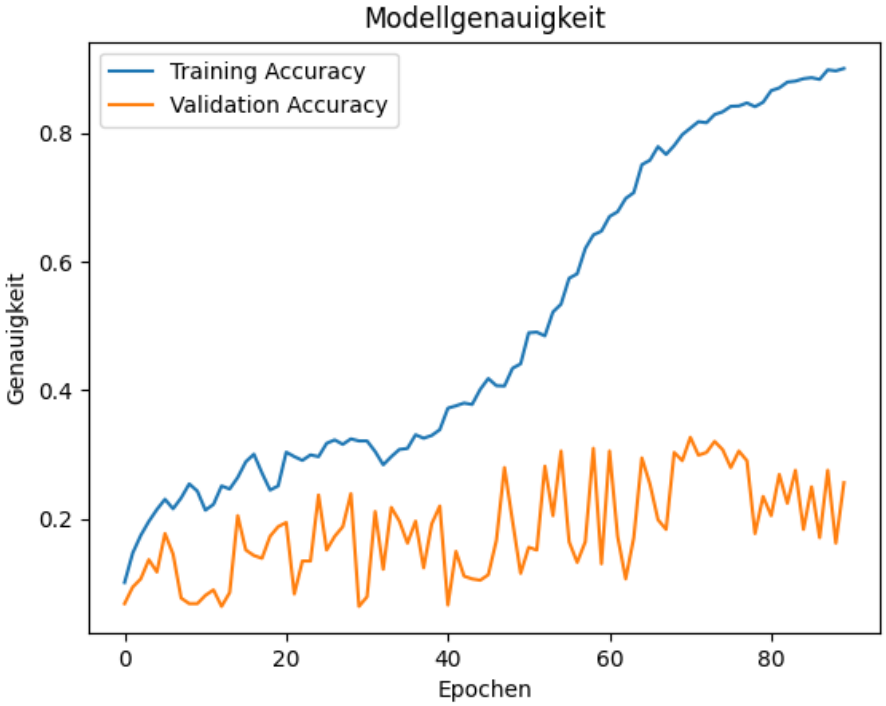


Abbildung 11 Accuracy des CNN über die Epochen

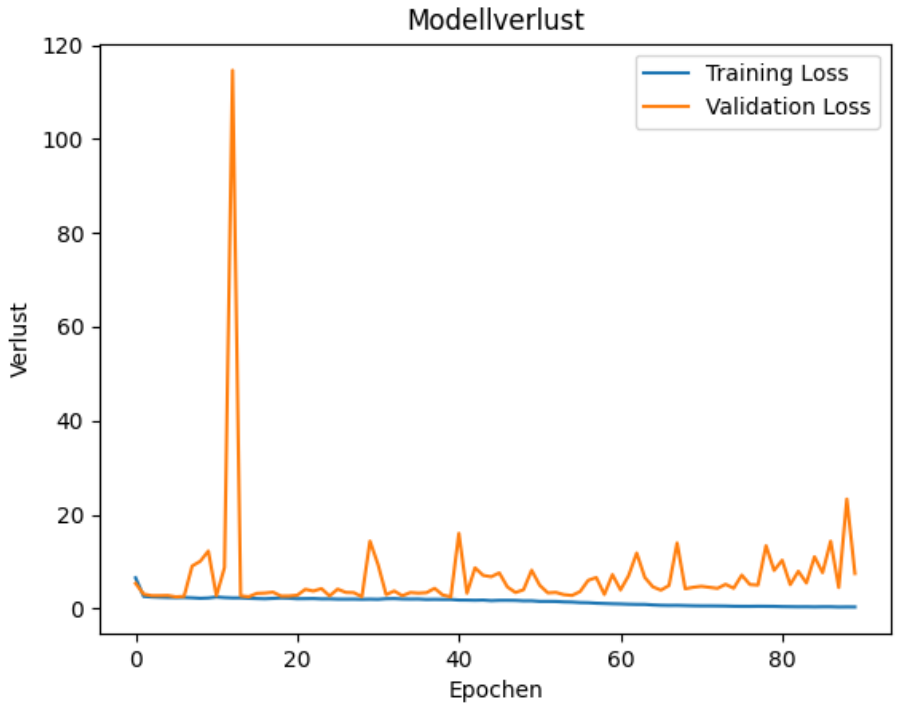


Abbildung 12 Loss des CNN über die Epochen

Aufbau des CNN:

- Conv2D mit Filteranzahl 16 und Filtergröße 1
- Conv2D mit Filteranzahl 16 und Filtergröße 3 (2x)
- Conv2D mit Filteranzahl 32 und Filtergröße 3 (3x)
- Conv2D mit Filteranzahl 64 und Filtergröße 3 (3x)
- BatchNormalization-Schicht
- Conv2D mit Filteranzahl 1 und Filtergröße 3
- Flatten-Schicht
- Dense-Schicht mit 32 Neuronen
- Dense-Schcht mit Anzahl der Neuronen = Anzahl der Klassen (15) und keiner Aktivierugsfunktion

Der Aufbau wurde nach umfangreichen Recherchen und einer Vielzahl von Trainingseinheiten mit verschiedenen Konfigurationen sorgfältig ausgewählt. Für alle Schichten mit einer Aktivierungsfunktion wurde die ReLU-Funktion aufgrund von vielversprechenden Ergebnissen in verschiedenen Tests als optimalste Funktion für das Zählen der Rittersporttafeln ermittelt.

3.5. Yolo 7

Tabelle 7 Evaluierung des Yolo 7

Kennzahl	Yolo 7
Accuracy	79,56 %
Precision	81,03 %
Recall	79,56 %
F1-score	79,13 %

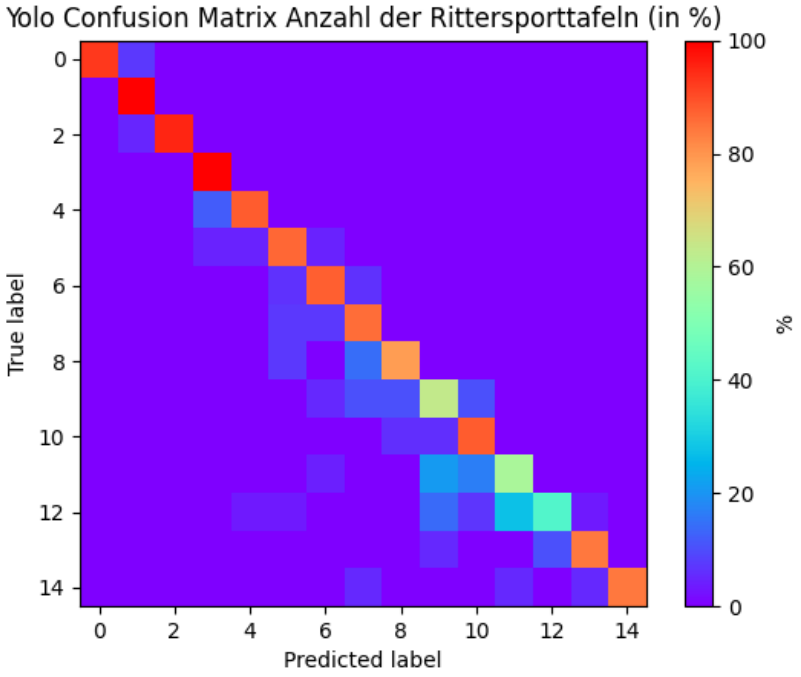


Abbildung 13 Confusion Matrix des YOLO 7

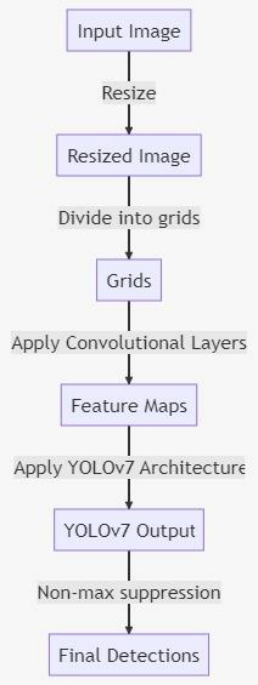


Abbildung 14 Ablaufgraph Yolo 7

4. Vergleich der KI-Verfahren

Bei der Evaluierung der Künstlichen Intelligenzen mit unterschiedlichen Verfahren ergeben sich deutliche Unterschiede in der Erfüllung des Ziels Rittersporttafeln zu zählen. Beim BN zeigt die selbstprogrammierte Implementierung eine leicht verbesserte Performance im Vergleich zum BN aus der scikit-learn Bibliothek. Die Unterschiede sind jedoch nicht signifikant. Beide BN-Verfahren zeigen ein gutes Ergebnis mit einem durchschnittlichen F1-score von ca. 50%. Die Confusion Matrix der BNs zeigen, dass die Randklassen mit wenigen und sehr vielen Rittersporttafeln deutlich besser erkannt werden und deutlich über dem Durschnitt liegen. In diesen Bereich erkennt die KI bis zu über 80% die richtige Anzahl an Rittersporttafeln auf dem Bild.

Beim KNN zeigt die selbstprogrammierte Implementierung eine leicht verbesserte Performance im Vergleich zum BN aus der scikit-learn Bibliothek. Wobei auch hier die Unterschiede nicht signifikant sind. Im Vergleich zu den anderen KI-Verfahren zeigt das KNN ein sehr schlechtes Ergebnis von unter 20% beim F1-score. In der Confusion Matrix ist zu sehen, dass die Prediction der KNN-KI nur für keine Rittersporttafeln gut ist. Das KNN zeigt bei seinen falsch-positiven Prediction keine Tendenzen.

Das scikit-learn RF-Verfahren zeigt deutliche bessere Ergebnisse als das selbstprogrammierte RF. Der F1-score des scikit-learn RF ist bei ca. 63% und ist somit acht Prozentpunkt besser als das selbstprogrammierte. Im Vergleich unter den klassischen KI-Verfahren hat das RF die besten Ergebnisse erzeugt. Diese guten Ergebnisse zeigen sich auch in der Confusion Matrix wieder und ist als eine deutliche Diagonale zu sehen. Insgesamt hat das RF die zweitbesten Ergebnisse erzielt.

Das CNN hat unter den gegeben Bilddaten keine guten Ergebnisse erzielt und kommt nur auf einen F1-score von lediglich 18% und ist somit das schlechteste betrachtete Verfahren. Eine mögliche Ursache für dieses Ergebnis ist, dass das CNN für die geforderte Aufgabe zu klein oder nicht über die passenden Layers verfügt. Des Weiteren besteht die Möglichkeit, dass die Datenanzahl zu gering oder zu unterschiedlich ist. Das CNN zeigt in der Confusion Matrix bei seinen falsch-positiven Predictions keine Tendenzen.

Das Yolo 7 liefert im Vergleich zu allen KI-Verfahren die besten Ergebnisse und erreicht einen F1-score von 79%. Die Confusion Matrix zeigt eine deutliche Diagonale und spiegelt den guten Ergebnissen in wahr-positive Predicitons wieder. Das Yolo 7 und CNN wurden mit den gleichen Datenpool trainiert und validiert, was zeigt, dass ein state of the art KI sogar mit stark unterschiedlichen Bilddaten gute Ergebnisse liefern kann.

Obwohl das YOLO-7-Modell im Vergleich zu den anderen KI-Verfahren bessere Ergebnisse erzielt hat, ist es wichtig, die Schwierigkeiten beim Training klassischer KI-Verfahren nicht zu vergessen. Bei klassischen Verfahren wie dem Random Forest (RF) und dem Bayesian Network (BN) hängt die Leistung stark von den Merkmalen (Features) ab, die den Modellen zur Verfügung gestellt werden. Wenn die Anforderungen an die Bilddaten genau definiert und konsistent sind, können sowohl das RF als auch das BN sehr gute Ergebnisse erzielen und sogar Industriestandards erfüllen. Dies bedeutet, dass eine sorgfältige Vorverarbeitung der Bilddaten und die Auswahl relevanter Merkmale entscheidend sind, um die Leistung dieser Modelle zu verbessern.

5. Anhang

A. Datenreihe zur Auswertung der klassischen KI-Verfahren

BN self

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	26,69	48,35	26,69	24,89
2	28,34	49,56	28,34	26,08
3	28,88	45,41	28,88	27,29
mean	27,97	47,77	27,97	26,09

BN bib

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	51,37	49,27	51,37	49,24
2	48,63	45,72	48,63	45,79
3	52,29	51,15	52,29	50,32
mean	50,76	48,71	50,76	48,45

BN self; FIP

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	50,64	51,08	50,64	49,56
2	52,65	51,75	52,65	51,35
3	53,2	49,66	53,2	50,71
mean	52,16	50,83	52,16	50,54

KNN self

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	17,92	19,47	17,92	18,42
2	17,92	19,47	17,92	18,42
3	17,92	19,47	17,92	18,42
mean	17,92	19,47	17,92	18,42

KNN bib

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	18,28	20,9	18,28	18,17
2	18,28	20,9	18,28	18,17
3	18,28	20,9	18,28	18,17
mean	18,28	20,90	18,28	18,17

RF self

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	54,3	57,27	54,3	54,42
2	53,93	55,07	53,93	54,1
3	58,32	59,58	58,32	58,46
mean	55,52	57,31	55,52	55,66

RF bib

Datenreihe	Accuracy	Precsion	Recall	F1-score
1	65,27	66,18	65,27	62,22
2	63,8	64,89	63,8	63,91
3	65,08	65,35	65,08	64,58
mean	64,72	65,47	64,72	63,57

B. Confusion Matrix Bayesian Network mit Merkmalsabhängigkeit