600 Jahre Universität Rostock

Traditio et Innovatio

Masterarbeit zum Thema

Erstellung und Evaluierung einer Datenpipeline zur automatisierten Klassifikation von Sammelkarten am Beispiel des Pokémon TCGs

Studiengang:	Wirtschaftsinformatik
Vorgelegt von:	Hauke Hansen Pruss
Matrikelnummer:	219202868
Bearbeitungszeitraum:	11. November 2024 – 31. März 2025
Erstgutachter:	Dr.-Ing. Hannes Grunert
Zweitgutachter:	M. Sc. Felix Hauptmann

Lehrstuhl für Wirtschaftsinformatik
Institut für Informatik
Fakultät für Informatik und Elektrotechnik
Universität Rostock

Inhalt

Kurzzusammenfassung Die vorliegende Arbeit beschreibt die Entwicklung einer Datenpipeline zur automatischen Klassifizierung von Pokémon-Sammelkarten. Ziel ist es, Verfahren aus der Literatur zur Vorverarbeitung und Klassifikation von Sammelkarten miteinander zu vergleichen, um eine effiziente und genaue Methode zur automatischen Erkennung und Klassifizierung der Karten zu entwickeln. Da Sammelkarten oft durch verschiedene Störeinflüsse, wie unterschiedliche Lichtverhältnisse aufgenommen werden, ist die Entwicklung einer robusten Methodik für deren eindeutige Erfassung wichtig. Dafür wurden Methoden der globalen und lokalen Merkmalsextraktion untersucht und miteinander verglichen. Der entwickelte Prototyp nutzt dafür eine Datenbank von Pokémon-Kartenbildern, um die Ansätze hinsichtlich ihrer Genauigkeit beim Bildabruf zu evaluieren. Die Ergebnisse zeigen, dass die helligkeitsbasierte Merkmalsextraktion in Verbindung mit einem trainierten Instanzsegmentierungsmodell in der Lage ist, eine hohe Erkennungsgenauigkeit zu erreichen. Mit zunehmender Komplexität an Störeinflüssen stößt die Methodik jedoch an ihre Grenzen. Die Arbeit stellt ein solides Grundgerüst für weitere Arbeiten im Bereich der Bildklassifikationsverfahren für Sammelkarten bereit und kann als Ausgangspunkt für weitere Experimente verwendet werden.

Kapitel 1

Einleitung

Das Hobby des Sammelns, Verkaufens und Tauschens von Sammelkarten begeistert Menschen aller Altersgruppen und hat eine lebendige Gemeinschaft von Enthusiasten geschaffen. Mit dem Aufkommen digitaler Technologien hat sich dieser Zeitvertreib erheblich weiterentwickelt: Sammler können sich über Online-Plattformen vernetzen, Informationen austauschen und Handel betreiben. Doch trotz dieser Fortschritte bleibt ein zentraler Aspekt des Sammelns problematisch – die manuelle Klassifikation und Registrierung von Karten. Für Sammler, die häufig mit großen Kartensammlungen arbeiten, ist es zeitaufwendig und mühsam, jede Karte manuell zu identifizieren und in eine Datenbank einzutragen. Die richtige Klassifikation erfordert oft spezifisches Fachwissen oder aufwendiges Nachschlagen in umfangreichen Datenbanken. Dies erschwert nicht nur die Verwaltung von Sammlungen, sondern kann auch den Wert einer Karte durch fehlerhafte Identifikation verfälschen. Die Popularität von Sammelkarten und der steigende Wert seltener Exemplare verstärken die Notwendigkeit automatisierter Tools. Bereits im Jahr 2010 lag der durchschnittliche Handelspreis für seltene Pokémon-Karten bei 7,56 Dollar[Bri10]. Hier setzt das Stralsunder Start-up Insleeve an, das innovative Lösungen zur Verbesserung der Benutzererfahrung im Sammelkartenhandel entwickelt. Ein zentraler Bestandteil ist die Entwicklung eines Kartenscanners, der es Sammlern ermöglicht, ihre Karten effizient und präzise zu klassifizieren, ohne diese manuell ins System eintragen zu müssen. Dieser Scanner soll relevante Informationen direkt aus den Karten extrahieren und eine eindeutige Klassifikation vornehmen. Diese Anforderungen bilden die Grundlage für die Entwicklung einer automatisierten Datenpipeline, die speziell auf die Klassifikation von Pokémon-Sammelkarten zugeschnitten ist. In dieser Arbeit wird eine solche Pipeline vorgestellt, die mithilfe verschiedener Algorithmen der Bilderkennung den gesamten Prozess von der Informationsgewinnung bis zur präzisen Klassifikation automatisiert und optimiert.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung und Evaluierung einer Datenpipeline zur automatisierten eindeutigen Klassifikation von Pokémon-Karten. Dabei liegt der Fokus auf der Verarbeitung von gescannten Bildern oder Aufnahmen, die unter nicht optimalen Bedingungen, wie ungenauer Ausrichtung oder ungleichmäßiger

Beleuchtung, entstanden sind. Die Pipeline gliedert sich in drei Hauptphasen. Zunächst wird die Karte durch eine präzise Segmentierung vom Hintergrund getrennt, um eine saubere Ausgangsbasis für die weiteren Verarbeitungsschritte zu schaffen. Im nächsten Schritt erfolgt die Extraktion relevanter Klassifikationsmerkmale der Karte. Abschließend werden diese Merkmale in einem Retrieval-Prozess mit einer Referenzdatenbank abgeglichen, um die ähnlichste Klasse zu ermitteln und eine eindeutige Identifikation der Karte zu ermöglichen. Dabei wird untersucht, welche Kombination aus Segmentierungs-, Informations- und Retrieval-Algorithmen gute Ergebnisse erzielt. Der Fokus liegt auf einer hohen Erkennungsgenauigkeit. Ziel ist es, auf Basis der in dieser Arbeit erzielten Ergebnisse Empfehlungen für eine Klassifikationspipeline abzugeben, die auch unter von Störungen belasteten Bedingungen eine zuverlässige und effiziente Klassifikation gewährleistet.

1.2 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in sieben Kapitel, die systematisch aufgebaut sind, um eine umfassende Analyse und Umsetzung der Konzepte zur Verarbeitung und Klassifikation von Sammelkartenspiel (SKS) Pokémon-Karten zu ermöglichen. In Kapitel 2 werden die grundlegenden Konzepte eingeführt, die für das Verständnis der Arbeit notwendig sind. Hierzu gehört eine Einführung in SKS Pokémon-Karten, die wesentlichen Aspekte der Bildverarbeitung sowie die Schritte der Vorverarbeitung (Preprocessing). In Kapitel 3 erfolgt ein Überblick über relevante aktuelle Forschungen und wissenschaftliche Arbeiten, die den Forschungsbereich näher eingrenzen. Dieser Abschnitt stellt Techniken und Ansätze vor, die für das Konzeptionskapitel von Bedeutung sind und einen Bezug zur aktuellen Forschungslage herstellen. Kapitel 4 widmet sich der Konzeption einer Datenpipeline für die Vorverarbeitung und Klassifikation der SKS Karten. Hier wird ein Experiment beschrieben, das die beste Kombination aus verschiedenen Klassifikationsverfahren in Verbindung mit Vorverarbeitungstechniken untersucht. Auf Basis eines Grundgerüsts als Ausgangsmodell wird zudem ein Testszenario konzipiert, um die verschiedenen Ansätze miteinander zu vergleichen. In Kapitel 5 wird die Umsetzung des Prototyps vorgestellt, der auf der in Kapitel 4 erarbeiteten Konzeption basiert. Dieses Kapitel enthält eine detaillierte Darstellung des verwendeten Codes, der Datenlage und der schließlich genutzten Pipeline, um die praktische Umsetzung anschaulich zu machen. Die Evaluation des entwickelten Prototyps erfolgt im sechsten Kapitel, in dem verschiedene Leistungsmetriken analysiert werden, die aus dem Testszenario abgeleitet wurden. Die Ergebnisse werden systematisch bewertet, um die Effektivität der Implementierung zu überprüfen. Abschließend werden in Kapitel 7 die wichtigsten Ergebnisse zusammengefasst und im Kontext der Forschungsfragen diskutiert. Darüber hinaus werden Vorschläge für weitere Forschungsansätze und mögliche Erweiterungen der Pipeline präsentiert, um zukünftige Entwicklungen im Bereich der Bildverarbeitung und Klassifikation von SKS Karten zu ermöglichen.

Kapitel 2

Grundlagen der Bildinformationsextraktion

Im folgenden Kapitel werden die grundlegenden Konzepte und Techniken erläutert, die für das Verständnis der Rahmenbedingungen dieser Arbeit von zentraler Bedeutung sind. Zunächst wird in Abschnitt 2.1 das Konzept des Bildabrufs vorgestellt, das den Kernbereich der Forschung absteckt und die damit verbundenen theoretischen Grundlagen vermittelt. Anschließend werden die Grundlagen der Bildverarbeitung behandelt, die eine zentrale Rolle bei der Analyse und Klassifikation von Bilddaten spielen. Im nächsten Schritt erfolgt eine Einführung in die Welt der SKS Pokémon-Karten, um die Bedeutung und die wesentlichen Merkmale dieser Sammelkarten zu erläutern. Besonders wird hierbei auf die Vorverarbeitung der Bilddaten eingegangen, die entscheidend für die Aufbereitung der Daten für die anschließenden Verfahren ist. Dabei werden sowohl klassische als auch moderne Vorverarbeitungsverfahren vorgestellt und deren Vor- und Nachteile diskutiert. Darüber hinaus werden verschiedene Bildklassifikationsverfahren erläutert, die es ermöglichen, die Karten anhand ihrer visuellen Merkmale zu kategorisieren. Diese theoretischen Grundlagen bilden die Basis für die weiteren Untersuchungen und Analysen in dieser Arbeit.

2.1 Themenabgrenzung anhand des Image Retrieval System

Bildabrufsysteme sind Technologien, die entwickelt wurden, um Bilder aus großen Datenbanken anhand ihrer Inhalte abzurufen. Diese Systeme können in drei Hauptaufgaben unterteilt werden: **Bildvorverarbeitung**, **Merkmalsextraktion** und **Bildvergleich**. Anhand des Literaturreviews von [MP16, S. 1f] fassten die Autoren die wichtigsten Schritte dieser Systeme zusammen. Ein Bildabrufsystem (Bas) besteht demnach aus drei methodischen Schritten, die nacheinander durchgeführt werden.

Bildvorverarbeitung: Im ersten Schritt werden alle Bilddaten für die weitere Analyse vorbereitet. Dazu gehören Verfahren wie Rauschreduktion, Umwandlung in besser verwertbare Farbräume. Ziel ist es ein standardisiertes Ausgabebild zu erzeugen, welches für jeden Durchlauf gleiche Ergebnisse liefert. Für diese Arbeit wichtige Verfahren werden in Abschnitt 2.3 vorgestellt.

Merkmalsextraktion: Ein nun standardisiertes Bild wird mittels verschiedener Verfahren auf besondere Bildmerkmale hin untersucht. Diese Merkmale werden zwecks Persistierung in einer Datenbank abgespeichert, um diese später abzurufen. Die Verfahren für die Merkmalsextraktion werden in Abschnitt 2.5 beschrieben.

Bildvergleich: Im letzten Schritt misst ein Bas die Ähnlichkeit zwischen einem abgefragten Bild und den Bildern in der Datenbank. Dies erfolgt durch den Vergleich der Merkmale des abgefragten Bildes mit den gespeicherten Merkmalen der Datenbank. Diesbezüglich typische Distanzfunktionen und Speicherlösungen werden in 2.5.3 vorgestellt.

In der in dieser Arbeit betrachteten Aufgabenstellung werden besonders großskalige Bildabrufsysteme betrachtet, bei denen Bilder aus sehr großen Datenbanken abgerufen werden. In der Forschung wird von großen Bildabrufsystemen ab einer Klassengröße von 10.000 einzigartigen Klassen gesprochen.

Die nachfolgenden Abschnitte des Grundlagenkapitels, befassen sich jeweils mit den drei Schritten des Bildabrufsystems, um die grundlegenden Verfahren vorzustellen.

2.2 Grundlagen der Bildverarbeitung

In diesem Abschnitt werden die Grundlagen der Bildverarbeitung erläutert. Diese grundlegenden Konzepte bieten das Fundament, auf dem die weiteren Themen aufbauen, und ermöglichen ein besseres Verständnis der Techniken und Algorithmen, die zur Analyse und Verarbeitung visueller Daten verwendet werden. Die Erzeugung eines Bildes durch das Abtasten einer Kameraaufnahme und die anschließende Quantisierung werden hier nicht näher erläutert, da sie den Rahmen des Themas überschreiten würden.

2.2.1 Bildrepräsentation und Formate

Bilder sind digitale Darstellungen visueller Informationen, die aus einer Sammlung von Pixeln bestehen. Jeder Pixel enthält Informationen zu seiner Farbe und Helligkeit und ist in einem Gitter angeordnet, das die Struktur des Bildes bildet. Es gibt verschiedene Arten von Bildrepräsentationen, abhängig von der Art der Information, die jedes Pixel speichert:

- Graustufenbilder: Hier wird für jedes Pixel ein Wert gespeichert, der die Helligkeit des Pixels angibt. Dieser Wert reicht normalerweise von 0 (schwarz) bis 255 (weiß), was in einem 8-Bit-Graustufenbild bedeutet, dass 256 unterschiedliche Helligkeitsstufen dargestellt werden können.
- Rot-Grün-Blau Farbraum: In einem RGB-Bild wird jeder Pixel durch drei Werte dargestellt, die die Intensität von Rot, Grün und Blau angeben. Diese Werte variieren typischerweise zwischen 0 und 255.

- HSV-Farbraum: Der HSV Farbraum ist ein Farbmodell, was auf der menschlichen Farbwahrnehmung basiert und dabei drei Hauptkomponenten umfasst. Der Farbton (hue) ist die Grundfarbe und wird in einem Wertebereich von 0° bis 360° dargestellt. Die verschiedenen Winkel beschreiben die Hauptfarben. Die Sättigung (Saturation) beschreibt die Intensität der Farbe in einem Wertebereich von 0 bis 100 %. Letzlich beschreibt die Helligkeit (value) die Reflektionskraft einer Farbe, beziehungsweise wie dunkel oder Hell diese ist [BGK15].

Die Farbtiefe eines Bildes gibt an, wie viele Farben pro Pixel dargestellt werden können. Eine höhere Farbtiefe ermöglicht eine genauere Darstellung von Farben, ist jedoch auch mit einer größeren Dateigröße verbunden. Beispielsweise hat ein 8-Bit-Bild 256 Farben, während ein 24-Bit-Bild 16,7 Millionen Farben darstellen kann. Graustufenbilder sind besonders für die Reduktion der Datenmenge und dementsprechender Vereinfachung von Algorithmen wichtig, wenn diese sich auf die Intensität und nicht auf die Farbe konzentrieren [Sac17].

Für die Persistierung eines digitalen Bildes haben sich einige Standardbildformate etabliert. Das Dateiformat PNG (Portable Network Graphics) ist für diese Arbeit am relevantesten. PNG ermöglicht eine verlustfreie Kompression und ist kompakt. Alle Bilder des Pokémon SKS liegen im PNG Format vor. Das Dateiformat kann bis zu 24 Bit Farbtiefe und somit 16,7 Millionen Farben unterstützen. Außerdem unterstützt es bis zu 16 Bit Graustufentiefe [Tan06].

2.2.2 Einführung in Pokémon Karten

Pokémon-Karten werden in Serien veröffentlicht, wobei jede Serie eine bestimmte Anzahl von Karten umfasst. Im Oktober 2024 zählte die Pokémon-Sammelkartenreihe mehr als 100 veröffentlichte Serien [bul24b], ergänzt durch über 100 Erweiterungen auf Basis der Grundserien [bul24a]. Jede Serie enthält dabei eine Vielzahl von Karten, die von einigen Dutzend bis hin zu mehreren Hundert reichen. Laut einer Schätzung sind derzeit 16.301 Pokémon-Karten im Umlauf [Aut24], und diese Zahl wird mit jeder neuen Veröffentlichung von Serien weiter steigen. Der Artikel von [mod19] bietet einen umfassenden Überblick über die verschiedenen Kartenarten, welche im Folgenden vorgestellt werden.

Die Gestaltung von Pokémon-Karten kann je nach Serie variieren, jedoch lässt sich jede Karte in drei Hauptkategorien unterteilen: Pokémon-Karten, Trainerkarten und Energiekarten. In dieser Arbeit werden jedoch ausschließlich die Pokémon-Karten betrachtet. Ein Pokémon kann aufgrund verschiedener Unterschiede in den Designmerkmalen und Abbildungen in mehreren Kartenserien erscheinen. In den folgenden Abschnitten werden die wichtigsten Merkmale, sowohl textuell als auch grafisch, erläutert.

Kartenmerkmale

SKS Karten sind standardmäßig genormt auf die Maße 63,5 mm x 88,5 mm. Eine Pokémon SKS kann durch die Kombination mehrere Kartenmerkmale in unterschiedlichen Paaren eindeutig identifiziert werden. Diese Merkmale umfassen den Namen des Pokémon, die Seriennummer, das Seltenheitssymbol und gegebenenfalls

spezielle grafische Effekte wie Hologramme oder Umkehr-Hologramme. Jedes dieser Merkmale besitzt eine spezifische Position auf der Karte, welches sich je nach Serie und Seltenheitsstufe der Karte unterscheiden.

Im oberen drittel einer Karte befindet sich der Name über dem Bild. Unter dem Bild befindet sich eine textuelle Beschreibung des Pokémon. Unter der textuellen Beschreibung werden bestimmte Fähigkeiten aufgelistet. Über die einzelnen Serien und Seltenheits-Evolutionsstufen verteilt befinden sich besondere visuelle Informationen, die allerdings nicht für die Identifikation der Karte dienen und dementsprechend ignoriert werden können.

Besonders hervorzuheben ist der Seltenheitswert einer Karte, der nicht nur ihren Wert bestimmt, sondern auch in Kombination mit der Seriennummer und dem Seriensymbol zur eindeutigen Identifikation einer Karte dient. Das Seltenheitssymbol befindet sich in der Regel am unteren Rand der Karte, direkt neben der Seriennummer. Dabei kann die Position des Symbols je nach Serie links oder rechts positioniert sein.

In Tabelle 2.1 werden die Seltenheitsstufen mit ihren jeweiligen Symbolen nach [mod19] erläutert.

Seltenheitsstufe	**Symbol**
Normal	●
Ungewöhnlich	♦
Selten	★
Seltene Holofolie	Sternform auf einem holografischen Design
Umgekehrte seltene Holofolie	Holoeffekt sichtbar auf dem Hintergrund
Sehr selten	Besondere grafische Gestaltung

Tabelle 2.1: Seltenheitssymbole im Pokémon SKS

Sehr seltene Karten zeichnen sich zusätzlich durch einzigartige Merkmale aus, die für die Klassifikation jedoch nicht von zentraler Bedeutung sind. Abbildung 2.1 zeigt die drei häufigsten Seltenheitswerte (Normal, Ungewöhnlich, Selten) anhand des Pokémon Gyarados:

(a) Normaler Seltenheitswert

(b) Ungewöhnlicher Seltenheitswert

(c) Seltener Seltenheitswert

Abbildung 2.1: Pokémon Gyrados mit unterschiedlichem Seltenheitswert

Aufgrund der Vielzahl von Merkmalen einer Karte ist eine eindeutige Identifikation durch nur einen Faktor schwierig. Abbildung 2.2 zeigt die möglichen Paarkombinationen, die zur eindeutigen Identifikation einer Pokémon-Karte notwendig sind. Diese Merkmale umfassen den Namen und die Seriennummer.

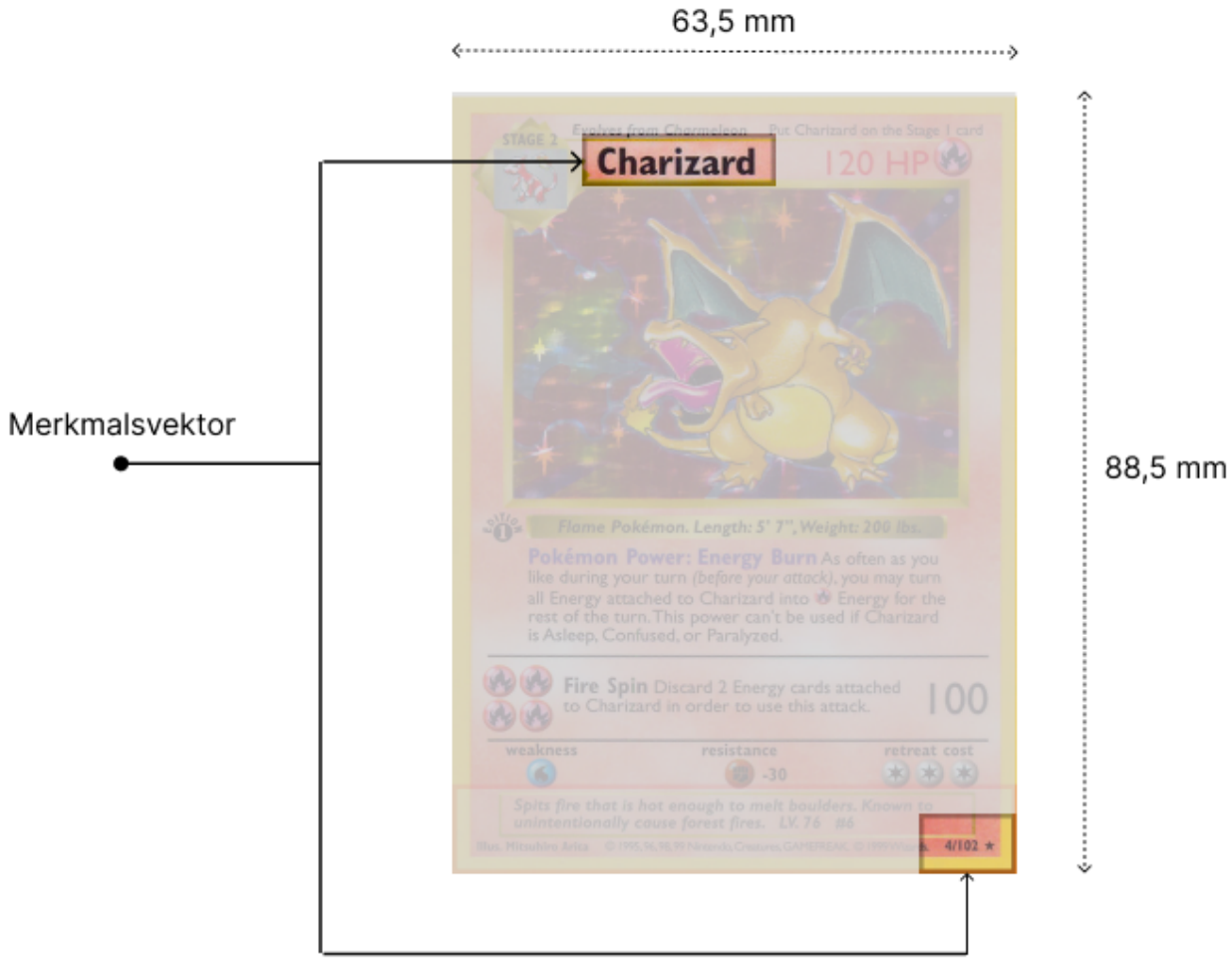


Abbildung 2.2: Möglicher eindeutiger Merkmalsvektor einer Pokémon Karte

Die Skalarvariablen eines möglichen eindeutigen Merkmalsvektors x können demnach definiert werden als:

$$\vec{x}_1 = \Big(\text{Name}, \quad \text{Seriennummer}\Big)$$

Andere Merkmalskombinationen sind möglich, aber nicht sinnvoll, da diese sich durch unterschiedliches Design in den Serien ändern und nicht eindeutig sind. Dies ist in Abbildung 2.1 zu erkennen. Abbildung 2.1 (a) hat im Gegensatz zu Abbildung 2.1 (b) keine Beschreibung. Abbildung 2.1 (c) wiederum hat an gleicher Stelle zusätzlich zur Beschreibung eine Sonderfähigkeit. Der erzeugte Merkmalsvektor ist interessant für die im Laufen dieser Arbeit verwendete inhaltsbasierte Klassifikation. Allerdings kann ebenfalls das gesamte Bild als globales Merkmal Verwendung finden.

2.3 Bildvorverarbeitungsverfahren

In diesem Abschnitt werden Methoden der Bildvorverarbeitung vorgestellt, die auf die Detektion und Isolierung einer spezifischen Pokémon Karte innerhalb eines gescannten Bildes abzielen. Dazu gehört die Trennung des Zielobjekts vom Hintergrund sowie die Korrektur von Rotationen, Verzerrungen und Farbabweichungen.

2.3.1 Rauschreduktion

Ein häufig auftretendes Problem von digitalisierten Bildern ist unerwünschtes Rauschen im Bild. Dieses ist meist charakterisiert durch randomisierte Variationen von ungewollter Helligkeit und oder zusätzlichen Bildinformationen. In Abbildung 2.3 werden drei typische Rauscharten abgebildet:

Abbildung 2.3: Typische Rauscheffekte

Gaußsches Rauschen entsteht durch zufällige Helligkeitsabweichungen, die einer normalverteilten Wahrscheinlichkeitsverteilung folgen. Salz- und Pfefferrauschen äußert sich in einzelnen Pixeln, die extrem hohe oder niedrige Werte annehmen, was zu auffälligen hellen oder dunklen Punkten im Bild führt. Punktrauschen bezieht sich auf isolierte Pixel, die stark von den benachbarten abweichen, und tritt ebenfalls als zufällige Störung auf [OIA19, S. , 389f]. Alle drei Rauscharten verursachen unerwünschte Bildverzerrungen, die die Bildqualität beeinträchtigen können. In der Forschung werden für die Entfernung dieser ungewollten Störeffekte verschiedene mathematische Filter diskutiert, welche in den folgenden Abschnitten vorgestellt werden.

Arithmetischer Filter

Der arithmetische Filter basiert auf dem Konzept der Mittelwertbildung innerhalb eines festgelegten Nachbarschaftsbereichs um jedes Pixel im Bild. Der Filter arbeitet, indem er den Durchschnittswert der benachbarten Pixel eines Bildes berechnet, wobei jedes Pixel durch den Mittelwert der Pixel in seinem Fenster ersetzt wird. Die mathematische Beschreibung des arithmetischen Filters, wie in [GW92, S. 344f] gegeben, lautet:

$$f(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

Hierbei wird der Ausgabewert $f(x,y)$ als Durchschnitt der Eingabepixelwerte $g(s,t)$ im Nachbarschaftsbereich S_{xy} um das Pixel (x,y) berechnet. Der Summenwert wird durch den Faktor $\frac{1}{mn}$ normalisiert, wobei m und n die Dimensionen des Fensters sind. Der Filter berechnet demnach den Durchschnitt aller benachbarten Pixel innerhalb eines Fensters der Größe $m \times n$. Dieser Durchschnitt wird dann als neuer Wert für das Pixel (x,y) im gefilterten Bild verwendet. Das Ziel des arithmetischen Filters ist es, Rauschen zu reduzieren und die Bildqualität zu verbessern, indem es die Variationen der benachbarten Pixel berücksichtigt [GW92, S. , 344].

Gaußfilter

Der Gaußsche Filter ist ein weit verbreiteter Filter in der Bildverarbeitung, der dazu dient, das Rauschen in Bildern zu reduzieren und Details zu glätten, ohne dabei wichtige Bildmerkmale zu stark zu verwischen. Der Filter basiert auf der Gaußschen Funktion, die als Gewichtsfunktion dient, um benachbarte Pixel in einem Bild unterschiedlich stark zu gewichten. Dabei haben Pixel näher am Zentrum des Filters einen größeren Einfluss auf den Ausgangswert als weiter entfernte Pixel. Mathematisch ausgedrückt ist der Gaußsche Filter eine Faltung (Convolution) eines Bildes mittels einer Filterbox. Die Faltung des Filters mit dem Bild bedeutet, dass jeder Pixelwert im Bild mit dem entsprechenden Wert des Filters kombiniert wird, um die lokale Änderung der Intensität zu berechnen. Der Filterkern $G(x,y)$ wird nach [Ron+14] im zweidimensionalen Raum durch folgende Formel gebildet:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Dabei ist σ die Standardabweichung der Verteilung, die die Breite des Filters steuert. Je größer der Wert von σ desto stärker wird das Bild geglättet. Dementsprechend glättet der Filter das Bild, indem er Pixelwerte auf Grundlage der benachbarten Pixel in einem bestimmten Bereich mittelt.

2.3.2 Bildgradienten

Diskrete Bildgradienten ermöglichen die Erkennung richtungsabhängiger Intensitätsänderungen in digitalisierten Bildbereichen. Diese Eigenschaft macht sie zu einem wichtigen Werkzeug in der Computergrafik und Bildverarbeitung, insbesondere bei der Detektion von Bildmerkmalen wie Kanten. Der Gradientenoperator wird in der Bildverarbeitung verwendet, um die räumlichen Änderungen der Bildintensität zu erfassen. Eine detaillierte Einführung in das Gradientenverfahren aus [MR09, S. 504-508] wird für die weitere Erläuterung

verwendet. In der kontinuierlichen Form eines Bildes $f_c(x, y)$ wird der Gradient durch die Ableitungen der Bildintensität nach den räumlichen Koordinaten x und y beschrieben und als Vektor dargestellt:

$$\nabla f_c(x, y) = \frac{\partial f_c(x, y)}{\partial x} i_x + \frac{\partial f_c(x, y)}{\partial y} i_y,$$

wobei i_x und i_y die Einheitsvektoren in der horizontalen und vertikalen Richtung sind. Der Betrag des Gradienten $|\nabla f_c(x_0, y_0)|$ misst die maximale Änderungsrate der Intensität an der Position (x_0, y_0). Der Gradient gibt sowohl die Stärke als auch die Richtung der Intensitätsänderung an. Da es sich bei den in dieser Arbeit betrachteten Bildern, um digitalisierte handelte, ist eine diskrete Betrachtung wichtig, bei der nur Intensitätswerte und Richtungen an festen Pixelpositionen definiert sind. In der diskreten Bildverarbeitung, bei der das Bild durch $f(n_1, n_2)$ repräsentiert wird (wobei n_1 und n_2 die Pixelkoordinaten in horizontaler und vertikaler Richtung sind), wird der Gradient durch diskrete Approximationen der Ableitungen des kontinuierlichen Gradienten berechnet. Diese Annäherung erfolgt durch Filter $h_1(n_1, n_2)$ und $h_2(n_1, n_2)$, die die Ableitungen in horizontaler und vertikaler Richtung repräsentieren und mit dem Bild gefaltet werden:

$$f_1(n_1, n_2) = f(n_1, n_2) * h_1(n_1, n_2), \quad f_2(n_1, n_2) = f(n_1, n_2) * h_2(n_1, n_2), \tag{2.1}$$

wobei $*$ die Faltung bezeichnet. Der geschätzte Gradient wird dann als Vektor definiert:

$$\nabla f(n_1, n_2) = f_1(n_1, n_2) i_{n_1} + f_2(n_1, n_2) i_{n_2}, \tag{2.2}$$

mit den Einheitsvektoren i_{n_1} und i_{n_2} in horizontaler bzw. vertikaler Richtung. Die Betragschätzung des Gradienten ergibt sich aus:

$$|\nabla f(n_1, n_2)| = \sqrt{f_1^2(n_1, n_2) + f_2^2(n_1, n_2)}, \tag{2.3}$$

während die **Richtungsschätzung** durch:

$$\theta(n_1, n_2) = \arctan\left(\frac{f_2(n_1, n_2)}{f_1(n_1, n_2)}\right) \tag{2.4}$$

berechnet wird. Diese Richtung gibt den Winkel der maximalen Intensitätsänderung an. Die Filter $h_1(n_1, n_2)$ und $h_2(n_1, n_2)$ können durch verschiedene Differenzoperatoren realisiert werden. Zwei der gängigsten Operatoren sind:

Erster Differenzoperator:

$$\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}$$

Dieser Operator berechnet die Differenz zwischen benachbarten Pixelwerten sowohl in der Horizontalen als auch in der Vertikalen.

Zentraler Differenzoperator:

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

Hier wird die symmetrische Differenz sowohl horizontal als auch vertikal berechnet.

Zur Verbesserung der Robustheit der Gradientenberechnung können diese Differenzen mit Glättungsoperato-

ren kombiniert werden. Beispielsweise verwenden erweiterte Operatoren wie der in Gleichung 2.5 vorgestellte Sobel-Operator größere Filtermasken, die sowohl die Differenzierung in einer Richtung als auch eine gewisse Glättung in der orthogonalen Richtung ermöglichen.

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{2.5}$$

Der Gradientenbetrag G an jedem Punkt im Bild kann dann berechnet werden als:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.6}$$

Dadurch werden die Rauschanfälligkeit verringert und die Gradientenberechnung robuster gegenüber Bildfehlern [MR09][S. 505]. In Abbildung 2.4 werden für einen gegebenen Bildausschnitt die Gradienten entlang der X- und Y-Achsen, der Gradientenbetrag sowie der Richtungsgradient anschaulich dargestellt.

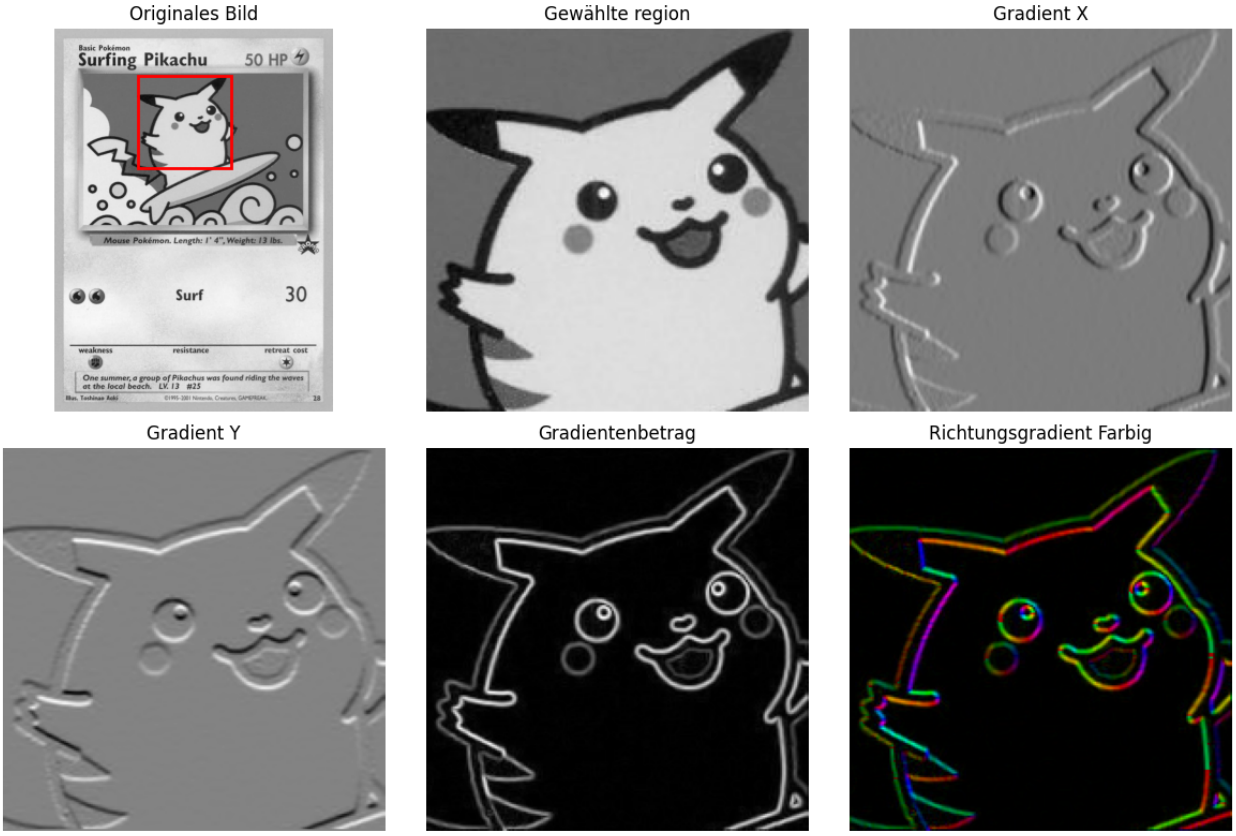


Abbildung 2.4: Darstellung der Schritte zur Gradientenberechnung

Die Gradientenrichtungen innerhalb des gewählten Bereichs werden gemäß der Farbkodierung in Tabelle 2.2 den entsprechenden Farben zugeordnet. Diese Kodierung dient allerdings lediglich für diskrete Bilder. Bei kontinuierlichen Bildern würde die Richtung in Grad ausgedrückt werden.

Farbe	Richtung (°)	Beschreibung
	0° / 360°	Positiv in x-Richtung
	90°	Positiv in y-Richtung
	180°	Negativ in x-Richtung
	270°	Negativ in y-Richtung

Tabelle 2.2: Farbzuweisung für Gradientenrichtung

Die in Abschnitt 2.3.1 vorgestellten Filter können diese Differenzen mit Glättung kombinieren, um die Rauschanfälligkeit zu verringern und die Gradientenberechnung robuster zu machen. Die Filtermasken der Verfahren verwenden größere Nachbarschaftsbereiche, die sowohl Differenzierung in eine Richtung als auch Glättung in die orthogonale Richtung ermöglichen.

2.3.3 Diskrete Kosinus Transformation

Dimensionalitätsreduktion

Die Extraktion von Bildmerkmalen erzeugt oft hochdimensionale Merkmalsräume, welche durch Ihre Komplexität oft keinen Zuwachs an Qualität für das Klassifikationsverfahren erzeugen. Daher ist die Wahl eines Verfahrens, welches die wichtigsten Merkmale aus einem Datensatz erfasst, wichtig. Dieser Abschnitt stellt das Kompressionsverfahren der Kosinustransformation vor, welches im Forschungsbereich des Bildabrufs häufig Verwendung finden. Abseits von statistischen Verfahren zur Dimensionalitätsreduktion, wie der Hauptkomponentenanalyse, werden insbesondere im Forschungsbereich der Bildabrufsysteme Verfahren aus der Signalverarbeitung verwendet. Die Diskrete Kosinustransformation ist ein Verfahren der Datenkompression, welches die Merkmale eines Bildes aus der räumlichen Domäne (Pixel) in Ihren Frequenzraum umwandelt. Im Kontext von Bildabrufsystemen wird es in der Forschung für die Reduktion von Histogrammen verwendet, da diese je nach Methodik eine hohe Dimensionalität aufweisen. Es gibt viele Definitionen der DCT Formeln, wobei hier der Fokus auf Definitionen für Bildanalyse gelegt wird. Die DCT Methodik wird daher im folgenden am Beispiel von Histogrammreduktion erläutert.

Ein Histogramm mit N Kategorien (Bins) kann durch die Funktion $f(t)$ beschrieben werden. Dabei gilt für die Werte von $t = 0, 1, \ldots, N-1$. Dabei repräsentiert $f(t)$ den Wert des Histogramm-Bins an der Stelle t.

Die Autoren von [Tan+18] nutzen dabei die Formel 2.7 für die Berechnung der eindimensionalen DCT für die Kompression von $f(t)$:

$$C(u) = a(u) \sum_{t=0}^{N-1} f(t) \cdot \cos\left(\frac{(2t+1)u\pi}{2N}\right), u = 0, 1, ..., N-1 \tag{2.7}$$

Wobei $a(u)$ definiert ist als:

$$\begin{cases} \sqrt{1/N}, & \text{if } u = 0, \\ \sqrt{2/N}, & \text{if } u = 1, 2, ..., N-1, \end{cases}$$

Für die Merkmalsextraktion werden meistens die niedrigfrequenten Koeffizienten ausgewählt, da sie die Hauptmerkmale des Bildes laut den Autoren bereits darstellen und stabiler gegenüber Rauschen sind. Diese ausgewählten Koeffizienten werden dann als Merkmale für die Bildverarbeitung verwendet.

2.3.4 Perspektivtransformation

Die Perspektivtransformation ist ein Verfahren, um ein Objekt, das aus einem schrägen Winkel fotografiert wurde, in eine frontale Ansicht zu überführen. Der erste Schritt in diesem Prozess ist die präzise Erkennung der vier Eckpunkte des Objekts. Für die Basis dieser Arbeit soll das Konzept anhand des Grundlagenwerks: „Multiple view geometry in computer vision“[HZ03, S. 32-36] erläutert werden. Betrachten wir das Trapez in Abbildung 2.5, das perspektivisch verzerrt dargestellt wird. Die Eckpunkte im verzerrten Bild werden wie folgt definiert:

$$P_1 = (x_1, y_1),\ P_2 = (x_2, y_2),\ P_3 = (x_3, y_3),\ P_4 = (x_4, y_4). \tag{2.8}$$

Um die Punkte als oben links, oben rechts, unten links und unten rechts zuzuordnen, werden die Koordinaten analysiert. Die Punkte mit den kleinsten y-Werten sind die oberen Punkte (P_1, P_2), und die Punkte mit den größten y-Werten sind die unteren Punkte (P_3, P_4). Innerhalb der oberen und unteren Punkte wird der Punkt mit dem kleineren x-Wert als „links“ und der mit dem größeren x-Wert als „rechts“ definiert. Nach dieser Zuordnung erhalten wir:

$$\text{oben links: } P_{ol},\ \text{oben rechts: } P_{or},\ \text{unten links: } P_{ul},\ \text{unten rechts: } P_{ur}. \tag{2.9}$$

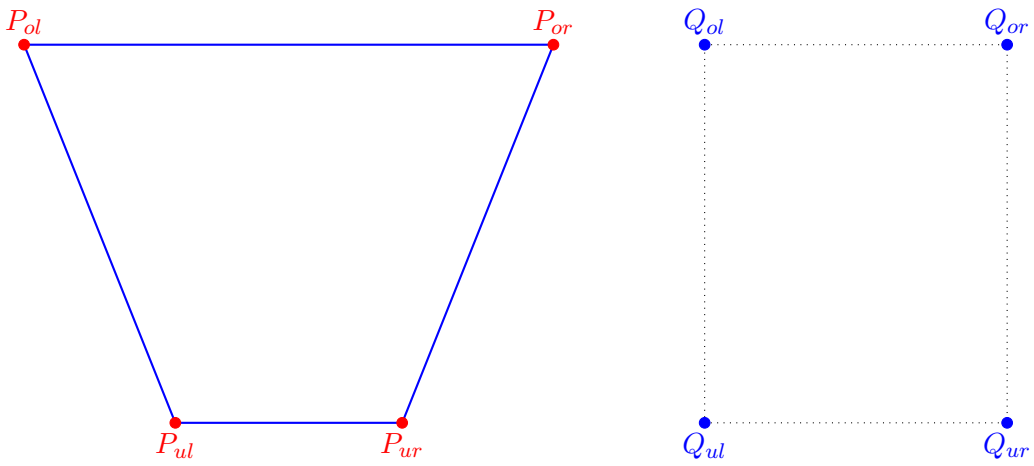


Abbildung 2.5: Verzerrtes Trapez P_i und Zielrechteck Q_i

Das Ziel der Perspektivtransformation ist es, das Trapez in ein Rechteck zu transformieren, dessen Eckpunkte durch die Zielkoordinaten definiert werden:

$$Q_1 = (0, 0),\ Q_2 = (w, 0),\ Q_3 = (0, h),\ Q_4 = (w, h), \tag{2.10}$$

wobei w die gewünschte Breite und h die gewünschte Höhe des Rechtecks sind. Die Ecken des Trapezes P_i werden dafür auf die entsprechenden Ecken des Zielrechtecks Q_i abgebildet. Für die Berechnung der Perspektivtransformation wird der Punkt $P = (x, y)$ im Bildraum durch homogene Koordinaten dargestellt. In homogener Form kann die Transformation eines Punktes auf einen neuen Punkt im Zielbild, nach [HZ03, S. 699], durch die folgende Gleichung beschrieben werden:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.11}$$

Hierbei bezeichnet (x, y) die Koordinaten des Punktes im Trapezbild, und (x', y') sind die transformierten Koordinaten im Zielrechteck. H ist die Homografiematrix, die die Perspektivtransformation beschreibt und als 3×3-Matrix gegeben ist:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{2.12}$$

Die Matrix H wird durch die Korrespondenz zwischen Punkten im ursprünglichen Bild und den gewünschten Zielpunkten berechnet. Für jeden Punkt $P_i = (x_i, y_i)$ im Trapez gibt es einen entsprechenden Punkt $Q_i = (x_i', y_i')$ im Zielrechteck. Für die Berechnung der Homografie nach [SS18, S. 699] müssen wir die folgenden Gleichungen aufstellen:

$$\begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11}x_i + h_{12}y_i + h_{13} \\ h_{21}x_i + h_{22}y_i + h_{23} \\ h_{31}x_i + h_{32}y_i + h_{33} \end{pmatrix} \tag{2.13}$$

Diese Gleichungen wird für alle vier Trapez-Ecken aufgestellt, wobei das Trapez in homogenen Koordinaten berücksichtigt wird.

Nachdem die Homografiematrix H berechnet wurde, kann sie auf alle Punkte im Trapez angewendet werden, um deren transformierte Positionen im Zielrechteck zu berechnen. Für jeden Punkt $P_i = (x_i, y_i)$ im Trapez ergibt sich der transformierte Punkt $Q_i = (x_i', y_i')$ im Zielrechteck durch die Anwendung der Matrix H:

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = H \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{2.14}$$

In Abbildung 2.6 ist eine durchgeführte Transformation mittels des hier vorgestellten Homografie Verfahrens dargestellt.

Abbildung 2.6: Durchführung des Homografieverfahrens

Es ist zu beachten, dass die Anwendung der Homografie auf ein Bild mit starker perspektivischer Verzerrung zu Qualitätsverlusten führen kann. Besonders bei sehr schrägen Perspektiven kann es zu Streckungen und Verzerrungen kommen, die die Bildqualität beeinträchtigen.

2.4 Instanzsegmentierung

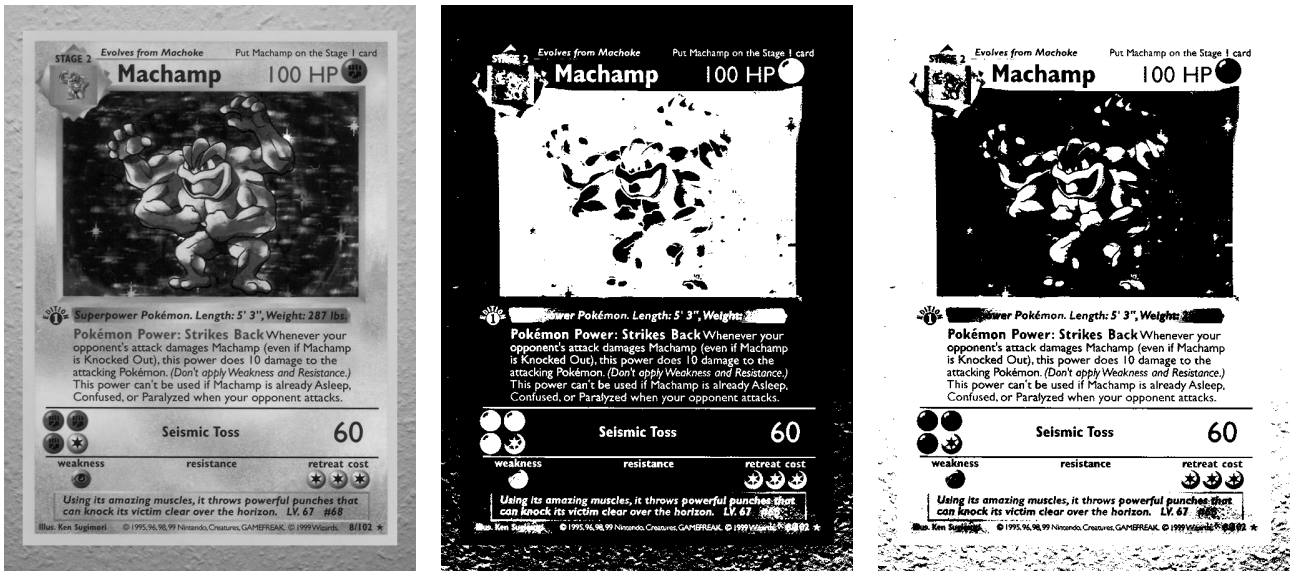
Die Segmentierung ist ein zentraler Bestandteil der Bildanalyse, da sie es ermöglicht, relevante Objekte vom Hintergrund zu unterscheiden. Insbesondere bei der Instanzsegmentierung wird nicht nur zwischen Objekten und Hintergrund differenziert, sondern jede einzelne Instanz eines Objekttyps eindeutig identifiziert. So können in einem Bild mehrere Pokémon erkannt und sicher voneinander abgegrenzt werden, wobei jedes Pokémon als eigene Instanz behandelt wird. Instanzsegmentierung bietet im Vergleich zur einfachen Objekterkennung einen entscheidenden Mehrwert, da sie pixelgenaue Masken für einzelne Objektinstanzen erzeugt. Dies ist insbesondere bei der Analyse spezifischer Objekte wie Pokémon-Karten hilfreich, da Kartengrenzen präzise erfasst und anschließend störende Hintergründe entfernt werden können. Diese Segmentierung bildet die Grundlage für weiterführende Schritte wie Perspektivkorrektur, Feature-Extraktion und die automatisierte Klassifikation mithilfe von Datenpipelines. In diesem Abschnitt werden die grundlegenden Methoden der Segmentierung basierend auf der Grundlagenliteratur von [You+04, S., 93–105] erläutert, darunter Schwellenwertverfahren und Kantenerkennungstechniken, die zur Identifikation von Objektgrenzen dienen. Anschließend werden in Abschnitt 2.4.3 moderne Ansätze der Instanzsegmentierung vorgestellt, die auf Convolutional Neural Networks (CNNs) basieren und eine deutlich höhere Präzision ermöglichen.

2.4.1 Schwellenwertverfahren

Das Schwellenwertverfahren ist eine Technik in der Bildverarbeitung, die verwendet wird, um Objekte vom Hintergrund zu unterscheiden, indem ein Helligkeitsschwellenwert q auf jedes Pixel im Bild angewendet wird. Der grundlegende Ansatz besteht darin, ein Pixel als Teil des Objekts zu klassifizieren, wenn seine Helligkeit das Schwellenkriterium erfüllt, andernfalls als Teil des Hintergrunds. Für helle Objekte auf dunklem Hintergrund gilt die Regel:

- Wenn $a[m, n] \geq q$, wird das Pixel an der Position (m, n) als „Objekt“ (1) markiert. Wobei a die Bildmatrix und m sowie n die Indizes des Pixels in der Höhe und Breite des Bildes beschreiben.
- Andernfalls wird es als „Hintergrund“ (0) markiert.

Für dunkle Objekte auf hellem Hintergrund kehrt sich die Regel um: Pixel mit $a[m, n] < q$ werden als „Objekt“ markiert. In Abbildung 2.7 wird der Schwellenwert anhand der binären und invertierten binären Schwellenwertgrenze für das Pokémon Vulpix dargestellt. Dabei wurde ein Schwellenwert von 127 gewählt, was bei einer 8-Bit-Darstellung 256 möglichen Graustufen entspricht. Dies bedeutet, dass alle Pixel mit einer Graustufenintensität größer oder gleich 127 als Objekt (weiß) und alle Pixel mit einer Intensität unter 127 als Hintergrund (schwarz) betrachtet werden. Dies gilt für die zweite Abbildung, invertiert. Wie zu erkennen, ist die Abgrenzung des Hintergrunds bei variierenden Schwellenwerten schwierig. Unterschiedliche Pokémon-Karten weisen eine Vielzahl an Farbtönen und Graustufen auf, die sich sowohl zwischen den Karten als auch zwischen den Hintergründen unterscheiden können. Daher ist es herausfordernd, einen universellen Schwellenwert zu finden, der in allen Szenarien eine zuverlässige Trennung zwischen Karte und Hintergrund ermöglicht. Ein niedrigerer oder höherer Schwellenwert kann die Segmentierung in manchen Fällen verbessern, jedoch ist er nicht in der Lage, alle Variationen der Karten und Hintergründe zuverlässig zu erfassen[You+04, S. 94-96].



(a) Originales Graubild (b) Binärer Schwellenwert (c) Seltener Seltenheitswert

Abbildung 2.7: Beispielbilder Schwellenwerte

2.4.2 Kantendetektion

Wie im vorherigen Abschnitt erläutert, dient die Schwellenwertberechnung der Segmentation aller Pixel, die zu einem Instanzobjekt gehören. Bezogen, auf in Ihrer Farbintensität stark variierender Bilder wie Pokémon Karten ist es aber selbst beim richtig gewählten Schwellenwert eine Herausforderung automatisiert zu erfassen, ob es sich beim segmentierten Bereich um die Karte oder ein anderes Hintergrundelement handelt. Methoden der Kantendetektion beruhen auf dem Prinzip, die Pixel des Grenzbereiches eines gegebenen Bildobjektes zu erkennen und vom Rest des Bildes zu isolieren. Dementsprechend könnten auch Intensitätsvariierende Instanzobjekte, wie Pokémon Karten vom Hintergrund abgegrenzt werden. Ein klassisches Verfahren der Kantendetektion ist der 1986 in [Can86a] vorgeschlagene Canny Algorithmus, dessen Schritte im Folgenden erläutert werden.

Rauschreduktion: Da Kantendetektion durch Rauschen beeinflusst werden kann, muss im ersten Schritt zunächst eine Rauschunterdrückung durchgeführt werden. Dafür verwendet Canny in seinem Ansatz[Can86b] einen Gaussischen Filter, dessen Funktionsweise bereits in Unterabschnitt 2.3.1 vorgestellt wurde.

Berechnung der Richtungsgradienten: Ein rauschreduziertes Bild kann mittels des in Unterabschnitt 2.3.1 vorgestellten Sobel Kernels in horizontaler (G_x) und vertikaler Richtung (G_y) gefiltert, um die erste Ableitung zu bilden. Dies ermöglicht, Kantengradienten und Richtungen für jedes Pixel zu finden.

Nicht-Maximum-Unterdrückung: Nach der Berechnung des Gradientenbetragsbildes $M[i, j]$ wird NMS angewendet, um die Kanten präzise zu positionieren und sicherzustellen, dass jede Kante nur ein Pixel breit ist. Der Canny-Algorithmus nutzt dabei ein 3×3-Nachbarschaftsfenster in acht Richtungen, um entlang der Gradientenrichtung zu interpolieren. Wenn $M[i, j]$ größer ist als die beiden interpolierten Werte in der Gradientenrichtung, wird es als Kantenpunkt markiert, andernfalls als Nicht-Kantenpunkt. Durch dieses Verfahren werden die möglichen Kandidaten gefunden.

Prüfung und Verbindung der Kanten: Der Canny-Algorithmus nutzt zwei Schwellenwerte nach dem in Unterabschnitt 2.4.1 vorgestellten Schwellenwerts verfahren, um Kantenpunkte zu bestimmen. Nach der Nicht-Maximum-Unterdrückung werden die Pixel entsprechend ihrer Gradientenstärke folgend klassifiziert:

- Liegt die Gradientenstärke eines Pixels über dem oberen Schwellenwert, wird es direkt als Kantenpunkt markiert.
- Liegt die Gradientenstärke unter dem unteren Schwellenwert, wird es als Nicht-Kantenpunkt betrachtet.
- Pixel mit Gradientenstärken zwischen diesen beiden Schwellenwerten werden als mögliche Kantenpunkte (Kandidaten) eingestuft.

Zum Schluss wird überprüft, ob diese Kandidatenpunkte mit bereits als Kantenpunkten markierten Pixeln verbunden sind. Ist dies der Fall, werden sie ebenfalls zu Kantenpunkten. Dieser Verbindungsprozess hilft dabei, unterbrochene Kanten zu vervollständigen und gleichzeitig Rauschen zu ignorieren, wodurch das finale Kantenbild klarer und weniger störanfällig wird [Ron+14, S. 578f].

(a) Originalbild (b) Gradientenstärke (c) Kantendetektion

Abbildung 2.8: Schritte des Canny Algorithmus

In Abbildung 2.8 werden die Schritte des Canny-Algorithmus visualisiert. Für das originale Bild in Abbildung 2.8a werden die Richtungsgradienten in Abbildung 2.8b berechnet. Im nächsten Schritt wird Nicht-Maximum Unterdrückung für die Positionierung der Kanten verwendet. Daraufhin wird die eigentliche Kantendetektion durchgeführt, dessen Ergebnis in Abbildung 2.8c dargestellt ist.

Es ist anzumerken, dass der Canny-Algorithmus lediglich eines von vielen Kantendetektionsverfahren ist, und als klassisches Verfahren hier zur Vorstellung des Grundprinzips dient.

2.4.3 Moderne Objekterkennung und -segmentierung

Die modernen Ansätze zur Objekterkennung und -segmentierung basieren maßgeblich auf tiefen neuronalen Netzwerken. Diese Netzwerke ermöglichen es, Objekte in Bildern nicht nur zu erkennen, sondern auch präzise zu lokalisieren. Dies beinhaltet auch die Segmentierung ihrer Grenzen. Eine der bekanntesten und effizientesten Architekturen in diesem Bereich ist das „You Only Look Once “(YOLO)-Modell, das Echtzeit-Objekterkennung mit hoher Genauigkeit kombiniert. Im Folgenden werden die grundlegenden Prinzipien der CNN-basierten Objekterkennung und Segmentierung erläutert. Dabei dient die YOLO-Architektur als exemplarisches Beispiel für Architekturen der modernen Instanzsegmentierung.

Die YOLO-Architektur (*You Only Look Once*), wie in [Red+16] vorgestellt, basiert auf einem einzelnen vollständig konvolutionalen neuronalen Netzwerk. Dieses Netzwerk verarbeitet ein Eingabebild in mehreren Schritten, um Objekte zu erkennen, ihre Positionen vorherzusagen und diese mit hoher Geschwindigkeit auszugeben. Die Autoren sahen die Objekterkennung als Regressionsprobleme, bei dem das Model direkt von den Pixeln des Bildes zu den Ausgabeparametern Bounding Boxes und den zugehörigen Klassenwahrscheinlichkeiten übergeht [Red+16, S. 1]. Abbildung 2.9 zeigt die Funktionsweise des Modells anhand des Beispiels aus dem originalen Paper [Red+16, S. 1f].

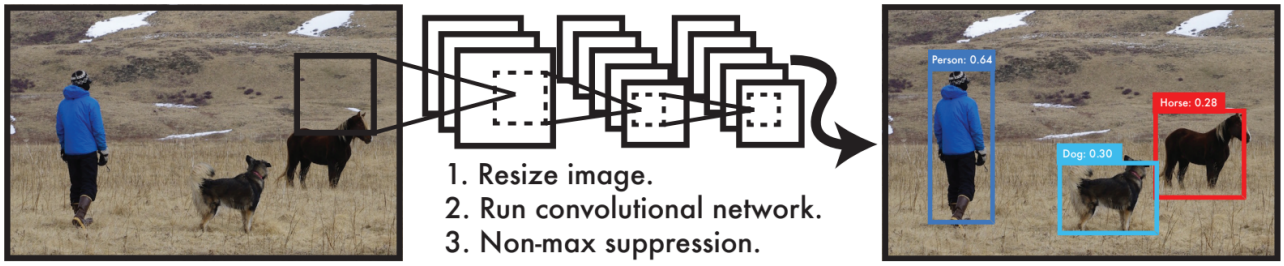


Abbildung 2.9: Yolo Detektions System [Red+16]

Die Funktionsweise lässt sich in die Schritte Vorverarbeitung, Merkmalsextraktion und Nicht-Maximum-Unterdrückung unterteilen, die im Folgenden vorgestellt werden.

Vorverarbeitung des Eingabebildes: Das Eingabebild wird zunächst auf eine feste Größe skaliert. Die Autoren verwenden hier 448×448 Pixel. Dieser Schritt dient dazu, alle Eingaben auf eine einheitliche Dimension zu bringen. Daraufhin erfolgt eine Normalisierung der Pixelwerte. Dies dient der Stabilisierung des Trainings und beschleunigt die Konvergenz. Das skalierte Eingabebild wird dann in ein gleichmäßiges Gitter mit einer Dimension $S \times S$ aufgeteilt. Dies ist im ersten Schritt der Abbildung 2.10 visualisiert. Jede Gitterzelle ist nun dafür verantwortlich, Objekte zu erkennen, deren Mittelpunkt in diese Zelle fällt.

Merkmalsextraktion: Die Basis des YOLO-Modells ist ein tiefes Convolutional Neural Network (CNN). Ein CNN besteht typischerweise aus mehreren Schichten. Folgend werden die wichtigsten vorgestellt:

- **Convolution-Schichten:** Diese Schichten wenden Nachbarschaftsfenster über das gesamte Eingabebild an, um Merkmale zu extrahieren. Jedes Nachbarschaftsfenster wird durch Parameter definiert, die während des Trainings gelernt werden.
- **Pooling-Schichten:** Diese reduzieren die räumlichen Dimensionen der Merkmalskarten. Dabei werden spezielle Werte innerhalb eines festgelegten Nachbarschaftsfensters ausgewählt werden.
- **Aktivierungsfunktionen:** Nach jeder Faltung wird eine Aktivierungsfunktion wie die *ReLU* (*Rectified Linear Unit*) angewandt. Diese führt Nichtlinearität ein. Die verhindert das, dass Netzwerk nur lineare Beziehungen lernt [Red+16, S. 3f]:

$$\text{ReLU}(x) = \begin{cases} x, & \text{wenn } x > 0 \\ 0, & \text{wenn } x \leq 0 \end{cases} \tag{2.15}$$

Bounding Boxes und Klassenwahrscheinlichkeits Berechnung: Jede Zelle des Gitters ist dafür zuständig, Objekte zu erkennen, deren Mittelpunkt in der jeweiligen Zelle liegt. Für jede Zelle sagt das Modell B Bounding-Boxes voraus. Diese Bounding-Boxes sind durch fünf Parameter definiert: (x, y, w, h). Es werden die Position, Weite und Höhe beschrieben. Zu jeder der generierten Bounding Box wird ebenfalls eine Konfidenzbewertung ausgeben.

Für jede Gitterzelle werden die folgenden Werte vorhergesagt:

- **Bounding-Box-Koordinaten:** Die Koordinaten (x, y) definieren, das Zentrum der Box relativ zur Gitterzelle, während w und h die Breite und Höhe relativ zum Bild angeben.
- **Konfidenzbewertung:** Diese bewertet, wie sicher das Modell ist, dass die Box ein Objekt enthält, multipliziert mit der Genauigkeit der Vorhersage. Eine Visualisierung der Bounding Boxes mit Konfidenzbewertung wird im oberen Bereich der Abbildung 2.10 gezeigt.
- **Klassenzugehörigkeit:** Jede Zelle gibt für jede Klasse eine bedingte Wahrscheinlichkeit vor, dass die Zelle einer Klasse angehört. Sobald die Wahrscheinlichkeit einen festgelegten Schwellwert überschreitet, gehört es der jeweiligen Klasse an. Eine so generierte Klassenkarte wird im unteren Bereich der Abbildung 2.10 gezeigt.

Nachbearbeitung: Nicht-Maximum-Unterdrückung (NMS): Nach der Vorhersage entfernt YOLO überlappende Bounding-Boxes mithilfe von Nicht-Maximum-Unterdrückung. Dessen Funktionsweise wurde in Unterabschnitt 2.4.2 vorgestellt. Dabei werden Boxen, die eine hohe Überlappung und eine geringere Konfidenzbewertung haben entfernt, um redundante Ergebnisse zu vermeiden [Red+16, S. 1]. Die finalen Detektionen werden auf der rechten Seite der Abbildung 2.10 visualisiert.

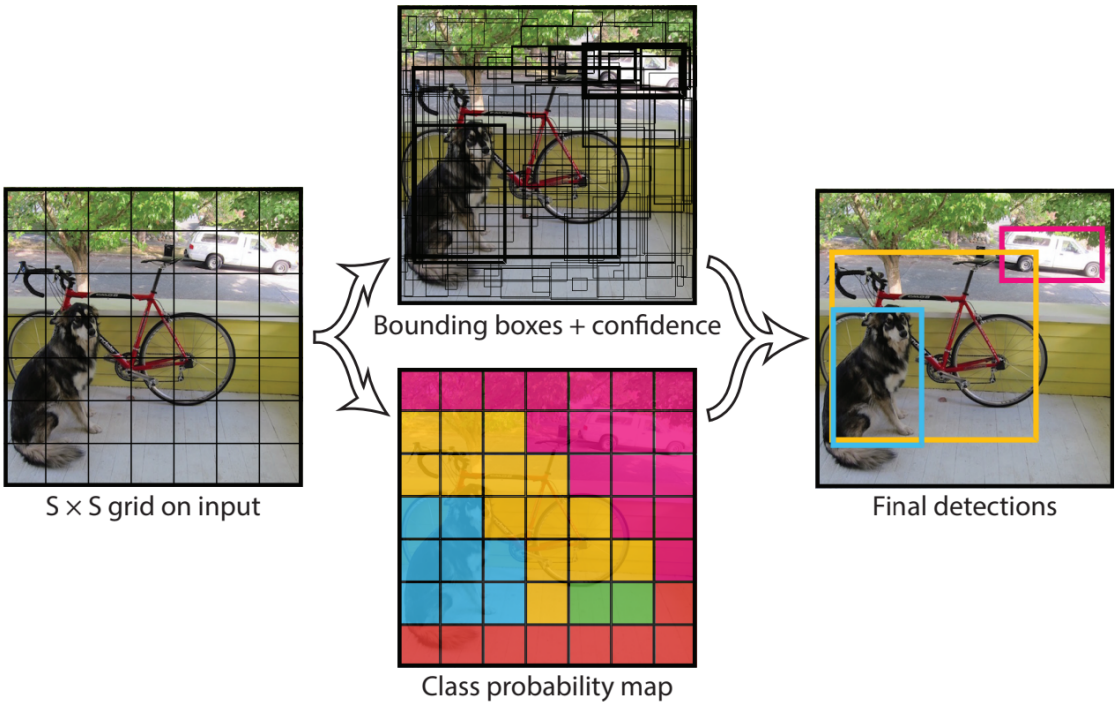


Abbildung 2.10: Yolo Model Bbox und Klassen Pipeline [Red+16]

YOLO integriert alle Schritte der Objekterkennung in ein einziges Netzwerk. Das bedeutet, dass das Modell direkt auf den Verlust für Objekterkennung optimiert wird[Red+16].

2.5 Merkmalsextraktion

Zur Extraktion von Informationen aus Bilddaten können verschiedene Verfahren zur Merkmalserkennung angewendet werden. Dabei werden die vorgestellten Verfahren auf den Forschungsbereich des Large Scale Image

Retrieval eingegrenzt. Die Autoren des Literaturreviews [Pin+13] teilen die Bildmerkmalsextraktion in zwei Teilbereiche, der globalen und lokalen Merkmalsextraktion ein. Für diese Teilgebiete werden die in dieser Arbeit verwendeten Verfahren in weiteren Verlauf des Kapitels vorgestellt.

2.5.1 Globale Merkmalsextraktion

Globale Featureextraktion beruht auf der Verwendung des gesamten Eingabebildes für die Extraktion von Merkmalen. In der Forschung werden verschiedenste Methoden diskutiert. Dieser Abschnitt stellt das HOG, Farbhistogramm, Perceptual Hashing Verfahren vor.

Histogram of Oriented Gradients

Das Histogram of oriented Gradients (HOG) ist eine klassische Methode zur Merkmalsextraktion. Zuerst in [DT05] als Methode für die Objekterkennung vorgestellt, kann das Verfahren auch für die Klassifikation verwendet werden. HOG extrahiert Kanten- und Forminformationen, indem es das Bild in kleine Zellen unterteilt. Es basiert auf dem gleichen Prinzip wie bereits in Unterabschnitt 2.3.2 vorgestellt. Der Gradient eines Bildes wird durch die Berechnung der Differenzen in den Pixelwerten entlang der horizontalen und vertikalen Achsen ermittelt. Für jede Zelle wird ein Histogramm der Richtungsgradienten erstellt. Das Histogramm enthält die Häufigkeit der Gradienten verschiedener Richtungen. Abbildung 2.11 stellt ein so entstehendes Histogramm an einem Beispiel dar. Zunächst wird ein Nachbarschaftsfenster variabler Größe über das Graustufenbild iteriert. Die Gradienten der jeweiligen Zellen werden berechnet und ein richtungsbasiertes Histogramm erstellt. Die entstehenden Histogramme werden zu einem globalen Merkmalsvektor zusammengeführt, der die Häufigkeit der verschiedenen Richtungen darstellt.

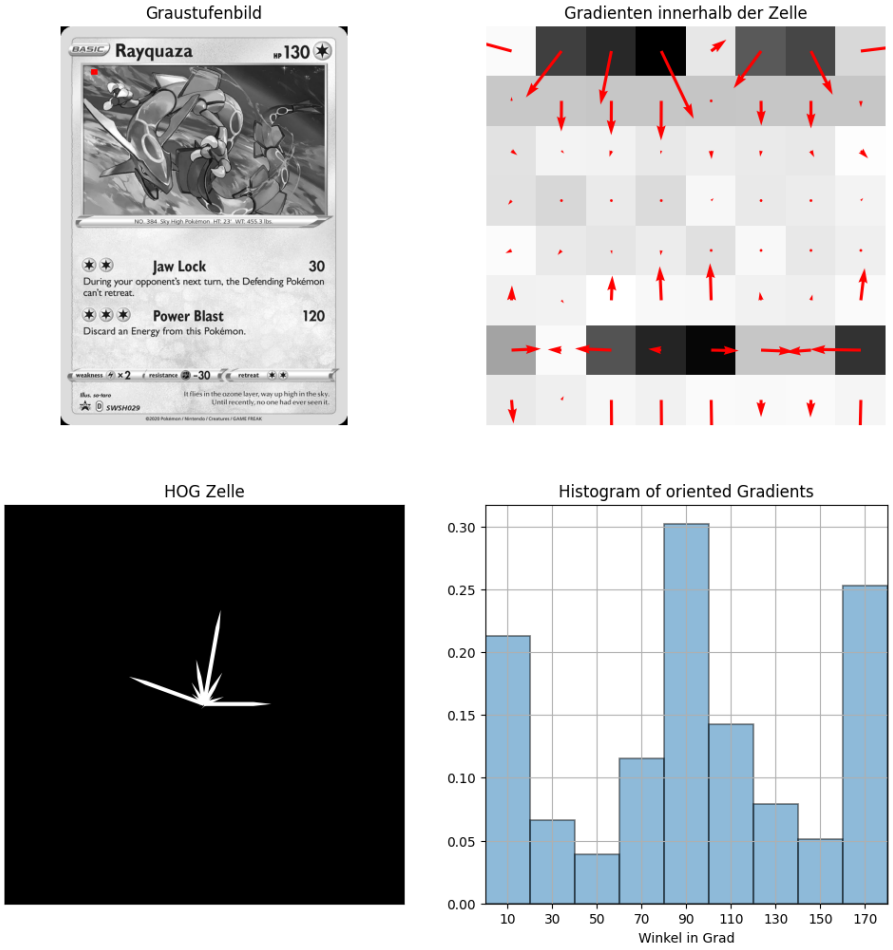


Abbildung 2.11: Darstellung der Schritte zur Berechnung nach HOG

In der Merkmalsklassifikation existieren in Bezug auf die Nutzung von Histogrammen weitere Verfahren. Das Verfahren Histogram of oriented Gradients konzentriert sich hauptsächlich auf strukturelle Informationen. Eine weitere Methodik ist die Farbhistogrammextraktion.

Farbhistogramme

Die Autoren von [Pin+13] stellen die Farbmerkmalsextraktion als klassische Methode vor, die auf Histogrammen basiert. Dabei wird die Häufigkeitsverteilung der Farben im Bild innerhalb eines Histogramms dargestellt. Diese Farb-Histogramme repräsentieren so das Bild durch die Häufigkeit der Farbwerte in unterschiedlichen Farbräumen. Die Autoren verwenden den RGB Farbraum. Dieser eignet sich besonders für Bilder, in denen das Histogramm der drei Farben eine Einzigartigkeit des Bildes gewährleistet. Einige Bildarten eignen sich allerdings nicht für die Verwendung von Farbhistogrammen im RGB Raum, da diese sich zu sehr ähneln. Daher wird in der Forschung häufig auch der HSV Farbraum für Farbhistogrammklassifikation verwendet. Farbhistogramme sind relativ robust gegenüber Transformationen wie Rotation und Skalierung. Abbildung 2.12 visualisiert die Histogrammbildung mittels dem RGB und HSV Farbraum.

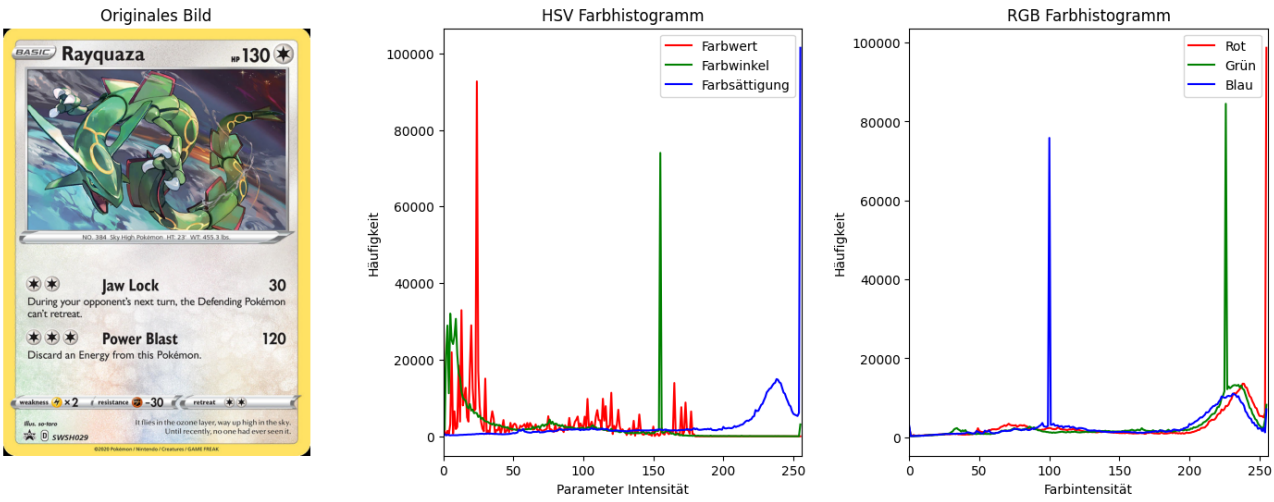


Abbildung 2.12: Histografische Visualisierung der Häufigkeitsverteilung im RGB und HSV Farbraum

Die Abbildung zeigt die Häufigkeitsverteilung (y-Achse) zur Intensität des jeweiligen Farbparameters (x-Achse). Im HSV Farbraum ist beispielsweise zu erkennen, dass der Farbwert zwischen der Intensität 23-27 eine sehr hohe Frequenz aufweist, sowie eine hohe Frequenz im Farbwinkel bei etwa 152 Grad aufweist. Im RGB Farbraum ist der Blauwert 100 sehr häufig vertreten, sowie der Rotwert 256 und Grünwert 225. Aus den daraus entstehenden Farbhistogrammen kann ein Featurevektor für die spätere Klassifikation erstellt werden.

Perceptual Hashing

Zusätzlich zur klassischen Farbextraktion werden in der Forschung Hash-basierte Verfahren untersucht. Dabei wird versucht die menschliche Bildwahrnehmung nachzuahmen. Das sogenannte Perceptual Hashing wandelt Bilder mit einem Hashing-Algorithmus in eine Signatur um, die spezifische Eigenschaften besitzt. Im Gegensatz zum klassischen kryptografischen Hashing bei den kleinen Änderungen in der Eingabe große Änderungen im Hash erzeugen, hat dies bei Perceptual Hash Algorithmen lediglich kleine Auswirkungen auf den erzeugten Hash. Kleine Änderungen im Bild führen dementsprechend nur zu minimalen Änderungen im Hash-Wert. Perceptual Hash-Algorithmen finden beispielsweise Anwendung bei der Google Bildersuche, um ähnliche Bilder zu einem Eingabebild zu finden.

Unabhängig vom spezifischen Algorithmus folgen Verfahren des Perceptual Hashings einem einheitlichen Ablauf, welcher sich meist wiederholt. Die Autoren von [Far21] bieten eine Übersicht relevanter Perceptual Hashing-Methoden, deren grundlegende Funktionsweise folgend vorgestellt wird:

1. **Vorverarbeitung:** Das Eingabebild wird auf eine einheitliche Resolution 300 x 300 Pixel konvertiert. Je nach verwendeten Algorithmus wird das Bild in einen benötigten Farbraum konvertiert.

2. **Hash Extraktion:** Aus dem gesamten Bild wird ein Hash berechnet. Die Autoren von [Far21] stellen dafür zwei verschiedene Verfahren vor. Zunächst wird das Eingabebild in 36 Blöcke unterteilt.

RGB-Farbraummethode: Aus diesen Blöcken wird aus jedem der drei RGB Farbkanäle ein Durchschnitt gebildet. Eine Umwandlung in Ganzzahlen ist nötig, um für jeden Block drei Werte zu erzeugen. Daraufhin wird der Hash konstruiert, indem alle 36 Werte aneinandergefügt werden. Daraus entstehen $36 * 3 = 108$ Werte, die bei 256 möglichen Farbwerten insgesamt $256^{108} \approx 1.23 * 10^{260}$ Hashwerte ermöglichen. Dieser mögliche Ergebnisraum verringert die Wahrscheinlichkeit einer Kollision zwischen Hashes.

Gradientenmethode: Hier wird das bereits in Abschnitt 2.3.2 vorgestellte Gradientenverfahren verwendet, um Farbunterschiede in horizontaler und vertikaler Richtung zu berechnen. Aus den 36 Blöcken wird der durchschnittliche Gradientenwert für jeden der drei Farbkanäle berechnet und anschließend ein Hash aus $36 * 3 = 108$ Werten generiert.

3. Hash Vergleich: Der erzeugt Hash kann im dritten Schritt mittels einer Distanzfunktion mit einem anderen Hash hinsichtlich seiner Ähnlichkeit verglichen werden.

Die in [Far21] vorgestellten Verfahren bieten einen guten Einblick, wie diese aufgebaut sind. Zusätzlich zu den genannten Verfahren existieren moderne Ansätze. Beispielweise verwenden Forscher tiefe neuronale Netzwerke, um Merkmale zu speichern und zur Klassifikation zu verwenden. Dieses Vorgehen ist jedoch für große Bildabrufaufgaben eher ungeeignet. Dies wird in Abschnitt 3.3 anhand relevanter Literatur weiter erläutert.

2.5.2 Lokale Merkmalsextraktion

Die Fokussierung auf das gesamte Eingabebild als Merkmalsvektor ist nicht für jede Klassifikationsaufgabe geeignet. Oft sind lokale Gegebenheiten eines Bildes bereits ausreichend, um das gesamte Bild anhand dieser Schlüsselpunkte identifizieren zu können. Dieser Abschnitt stellt zu diesem Zwecke die Methodik der lokalen Texterkennung, sowie das Local Binary Pattern vor.

Local Binary Patterns

Statt die Häufigkeit der Farbe eines Bildes zu messen, konzentriert sich die Local Binary Pattern (LBP) Methodik auf die Texturinformationen von lokalen Nachbarschaftsfenstern des Bildes. Die Autoren von [HPS09] untersuchten die Methodik in Bezug auf die Erkennung von Schlüsselpunkten in Bildern. Die Methodik ist besonders interessant, da sie relativ robust gegenüber Änderungen des Helligkeitswertes eines Bildes ist. Diese Eigenschaft beruht darauf, dass LBP die relative Intensität der Nachbarschaftspixel, statt des absoluten Wertes messen. Der Berechnungsprozess eines Merkmalsvektors mittels der LBP-Methode wird in Abbildung 2.13 anhand des Papers [Mül+20] anschaulich vorgestellt.

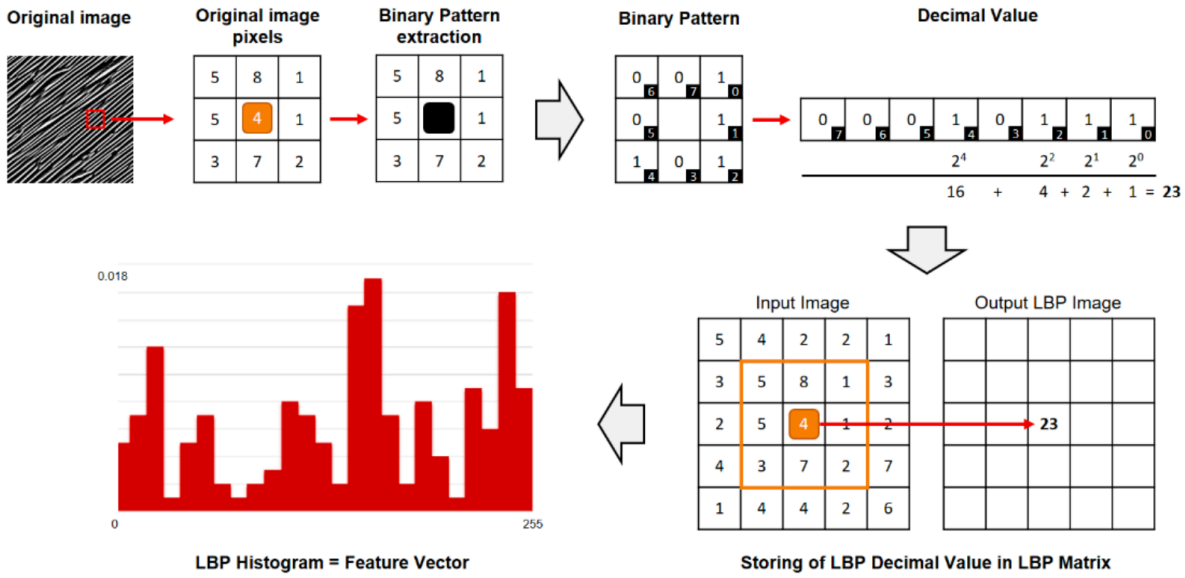


Abbildung 2.13: Berechnung eines Featurevektors mittels Local Binary Pattern [Mül+20]

Ein Eingabebild wird zunächst mittels eines Nachbarschaftsfensters variabler Größe durchiteriert. Jede Iteration des Nachbarschaftsfensters berechnet das Grauwertverhältnis des lokalen Nachbarschaftsfensters. Dabei wird der Grauwert des zentralen Pixels n_c mit den Grauwerten der Pixel der Nachbarn n_i verglichen. Für die Berechnung des Binary Pattern gilt: Wenn $n_i >= n_c$ wird dem Bereich eine 1 zugewiesen. Ansonsten wird eine 0 gesetzt.

Die daraus resultierende Binärmatrix wird als N-Bit-Zahl interpretiert. Dadurch sind 2^N Muster möglich. Nach der Berechnung des LPB Dezimalwertes kann ein Histogramm mit 2^N Kategorien erstellt werden, wobei N die Anzahl der Nachbarn des Mittelpunkts ist. Bei 8-Bit Grauwerten ergeben sich $0 - 255$ mögliche Kategorien. Die berechneten Dezimalwerte werden Ihren Kategorien im Histogramm zugeordnet und so die Häufigkeit dieser inkrementiert [HPS09].

OCR-Verfahren

Im Hinblick auf zu extrahierende Merkmale aus Bildern und besonders Dokumenten beschäftigt sich die Forschung schon lange mit Methoden der Texterkennung. Diese als Optical Character Recognition (OCR) bezeichneten Systeme basierten anfangs auf regelbasierten Algorithmen. Dabei wurden Zeichen anhand von geometrischen Analysen erkannt. Diese Verfahren waren jedoch empfindlich bei Variationen der Schriftart oder der Textgröße. Durch die Einführung stochastischer Modelle wie dem Hidden Markov-Modell konnten Sequenzen besser erkannt und so auch handgeschriebene Texte erfasst werden. Durch die Verbreitung von Deep Learning und besonders den darauf basierenden Long Short Term Memory (LSTM) Netzen war die Verarbeitung langer Texte mit hoher Erkennungsgenauigkeit möglich.

Dabei basieren moderne Texterkennungspipeline, wie von den Autoren in[ZTB20] vorgestellte, auf dem erstmalig in [HS97] eingeführten LSTM Prinzip. Die Texterkennung mittels LSTM erfolgt anhand mehrere Schritte, die der Textextraktion dienen. Die Texterkennung mit Long Short-Term Memory (LSTM) erfolgt in mehreren Schritten, die darauf abzielen, Bilddaten zu analysieren und den enthaltenen Text zu extrahieren.

Bildvorverabeitung: Im ersten Schritt wird das Bild in Graustufen umgewandelt und normalisiert. Damit kann der relevante Textbereich mittels Segmentierung extrahiert werden. Dabei werden ähnliche Verfahren, wie die in Abschnitt 2.9 vorgestellte Vorverarbeitung des Eingabebildes verwendet.

Merkmalserkennung: Aus den segmentierten Bereichen wird unter Verwendung der bereits in Abschnitt 2.4.2 vorgestellten Kantendetektion möglicher Text extrahiert. Die Auswahl der Kandidaten erfolgt anhand von Schwellenwerten, dessen Funktionsweise bereits in Abschnitt 2.4.1 vorgestellt wurde.

LSTM Verarbeitung: Die Textkandidaten werden nun vom LSTM Modell verarbeitet. Dabei ermöglichen die Speicherzellen des LSTM die extrahierten Kandidaten auf Ihre Relevanz hin zu untersuchen. Das Modell analysiert so die Kandidaten auf Ihre mögliche Zeichenbedeutung und gibt das wahrscheinlichste Zeichen aus Ausgabe aus. Im Anschluss werden die Zeichen zu einem vollständigen Satz zusammengefügt[HS97, S. 9f].

2.5.3 Ähnlichkeitsberechnung

Die in Abschnitt 2.5 vorgestellten Merkmalsvektoren eines Eingabebildes werden in der Forschung unterschiedlich persistiert. Um eine Ähnlichkeitsberechnung zwischen zwei Bildmerkmalsvektoren desselben Datentyps durchzuführen, werden Distanzberechnungsmethoden benötigt. Die Autoren von [PT12] stellen die wichtigsten Methoden vor, welche im Folgenden näher erklärt werden.

Bei der Ähnlichkeitsberechnung zweier Bilder gilt, dass je kleiner der berechnete Distanzwert ist, desto ähnlicher sind die Bilder. Für die Berechnung dieses Distanzwertes kommen für unterschiedliche Methoden unterschiedliche Distanzmetriken infrage. Falls nach [PT12, S. 154-159] x und y zwei d-dimensionale Featurevektoren der Datenbank und Abfragebildes sind, kann die Ähnlichkeit berechnet werden mittels:

Euklidische Distanz

Misst den geradlinigen Abstand zweier Punkte:

$$d_{\mathrm{E}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2.16}$$

Manhattan-Distanz

Summiert die absoluten Differenzen der Koordinaten:

$$d_{\mathrm{MAN}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|. \tag{2.17}$$

siehe [PT12, S. 154-159]

Zusätzlich zu diesen klassischen Distanzmetriken von den Autoren von [Sur+03] einerseits die Vektor-Cosinus Grad-Distanz, sowie die CHI-Quadrat-Distanz vorgestellt.

Vektor-Cosinus-Grad-Distanz

Die Cosinus-Distanz zwischen zwei Vektoren $\mathbf{x}$ und $\mathbf{y}$ wird definiert als:

$$\text{VCAD}_{\cos}(\mathbf{X}, \mathbf{Y}) = 1 - \frac{\mathbf{X} \cdot \mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|},$$

wobei $\mathbf{x} \cdot \mathbf{y}$ das Skalarprodukt ist. Der Ausgaberaum von $VCAD \in [0, 1]$ ermöglicht einen einfachen Vergleich anhand der Ähnlichkeit.

Chi-quadrat-Distanz

Die Chi-quadrat Distanz kann für den Vergleich der Häufigkeit der einzelnen Kategorien zweier Histogramme verwendet werden. Die Anzahl der Kategorien der beiden Histogramme muss dabei gleich sein.

$$\text{d}_{\text{CHI}}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i} \tag{2.18}$$

Hamming-Distanz

Die Hamming-Distanz wird für binäre Vektoren verwendet. Sie gibt die Anzahl der Positionen an, an dessen sich die entsprechenden Elemente unterscheiden:

$$d_{\text{ham}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|, \tag{2.19}$$

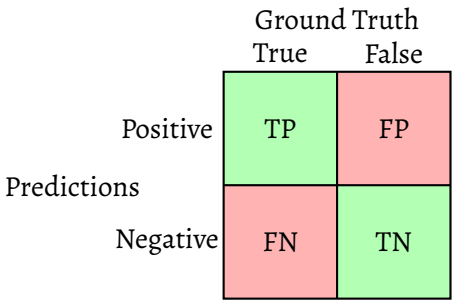
wobei $x_i, y_i \in \{0, 1\}$.

Levenshtein-Distanz

Die Autoren von [AA21] beschreiben die Levenshtein-Distanz $d(ab, b)$ als die minimale Anzahl an Operationen, die nötig sind, um einen Zeichstring a in einen anderen b umzuwandeln. Dafür werden die Operationen Einfügen, Löschen und Ersetzen von Charakteren verwendet. Falls einer der Strings leer ist, entspricht die Distanz der Länge des anderen Strings. Ansonsten wird die Distanz aus der Anzahl an Änderungen, die durch die drei Operationen nötig sind, um den einen String in den anderen zu überführen, berechnet.

2.6 Leistungsmetriken

Für die Evaluation von Klassifikationsaufgaben sind Metriken von Vorteil, welche einen effizienten und automatisierbaren Vergleich verschiedener Klassifikationsmethoden ermöglichen. Im Forschungsbereich der Bildklassifikation haben sich die Metriken Präzision, Sensitivität und der F-1 Score durchgesetzt. Um diese Metriken zu erläutern, muss zunächst das Konzept der Konfusionsmatrix vorgestellt werden. Die Erläuterung der Konfusionsmatrix, sowie die Definition der Metriken, erfolgt anhand [Bur19][S.15]. Eine Konfusionsmatrix ist eine Tabelle, welche den Erfolg einer Klassifikationsaufgabe misst. Eine Achse des Models enthält das vom Klassifikationsmodell vorgeschlagene Merkmal (Prediction), während die andere Achse das echte Merkmal (Ground Truth) repräsentiert.



Legende: TP = True positive; FP = False positive; TN = True negative; FN = False negative

Abbildung 2.14: Konfusionsmatrix:

Präzision ist das Verhältnis der korrekten positiven Vorhersagen zur Gesamtzahl der positiven Vorhersagen. Damit kann überprüft werden, wie präzise ein Modell positive Vorhersagen tätigt.

$$Precision = \frac{TP}{TP + FP}$$

Sensitivität ist das Verhältnis richtiger positiver Vorhersagen zur Gesamtzahl positiver Beispiele im Testset. Es eignet sich, um Klassifizierungsmodelle auf Ihre Fähigkeit positive Ergebnisse zu erkennen zu testen.

$$Recall = \frac{TP}{TP + FN}$$

Der F1-Score ist eine weitere Evaluationsmetrik, die Sensitivität und Präzision kombiniert, um eine kombinierte Metrik für die Modellbewertung zu schaffen. Die Kombination bildet ein harmonisches Mittel. Es eignet sich als Metrik besonders für Klassifizierungsaufgaben, bei denen von Sensitivität und Präzision ein hohes Niveau erwartet wird [Sun+22][S.9].

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

Die Präzision und die Sensitivität eignen sich zusätzlich für die Erstellung grafischen Leistungsmetrik. Die PR-Kurve (Precision-Recall-Kurve) ist eine Grafik, die Präzision und Recall gegenüberstellt, wobei Präzision die vertikale und Recall die horizontale Achse darstellt. Sie wird häufig im Bereich der Informationsextraktion genutzt, um den Anteil der tatsächlich positiven Proben zu visualisieren [Sun+22][S.10].

Kapitel 3

Verwandte Arbeiten / Stand der Technik

Im folgenden Kapitel wird die relevante Literatur zu bestehenden Bildabrufsystemen, dazugehöriger Bildvorverarbeitung, sowie Bildmerkmalsextraktion vorgestellt. Dabei wird insbesondere auf verwandte Forschungsergebnisse eingegangen, die in diesen Bereichen erzielt wurden. Diese dienen als Grundlage, um die aktuellen Entwicklungen und Herausforderungen zu analysieren. Auf Basis dieser Erkenntnisse wird in Kapitel 4 eine konkrete Zielsetzung für die vorliegende Arbeit herausgearbeitet, die auf den identifizierten Forschungslücken und offenen Fragestellungen aufbaut.

In der ersten Iteration wurden die Suchmaschinen Scopus und Google Scholar genutzt, um nach grundlegenden Arbeiten zu Bildabrufsystemen zu suchen. Die Suchbegriffe waren:

- „Bildabrufsysteme“
- „image retrieval systems“

Es wurden Paper ausgewählt, die eine vollständige Pipeline als Systemansatz vorstellten.

Bildvorverarbeitung

Für die Bildvorverarbeitung wurden die folgenden Suchbegriffe verwendet, um relevante Literatur zu identifizieren:

- „Bildvorverarbeitung“
- „Image Preprocessing OR Segmentation“

Die ausgewählte Literatur musste das später in diesem Kapitel besprochene Hintergrundproblem für Karten lösen, um in die engere Auswahl zu gelangen.

Merkmalsextraktion

Bei der Suche nach Literatur zur Merkmalsextraktion, inspiriert durch die Ergebnisse der Bildabrufsysteme, wurden die folgenden Suchstrings verwendet:

- „image retrieval techniques“
- „image feature extraction“
- „image retrieval review“
- „content-based image retrieval“

Die identifizierten Arbeiten wurden hinsichtlich der verwendeten Merkmalsextraktionsmethoden und insbesondere der Anzahl der enthaltenen Datenbanksätze analysiert, um einen breiten Überblick über die möglichen Methoden zu erhalten.

Distanzmetriken

Zur Untersuchung der Distanzmetriken, die in Verbindung mit Bildabrufsystemen stehen, wurden die folgenden Suchbegriffe verwendet:

- „image retrieval distance metrics“
- „image retrieval distance similarity“

Bei der Suche nach relevanten Distanzmetriken wurde gezielt nach Studien gesucht, die die Kombination von Bildabrufsystemen mit Distanzmetriken oder Ähnlichkeitsmaßen untersuchten.

3.1 Bestehende Bildabrufsysteme

Die inhaltsbasierte Bildsuche Content based image retrieval (CBIR) kombiniert verschiedene Disziplinen. Einerseits spielen Bildvorverarbeitung und Merkmalsextraktion eine zentrale Rolle, um Bilder so zu beschreiben, dass zwischen ähnlichen und unähnlichen Bildern unterschieden werden kann. Andererseits geht es um die Speicherung und effiziente Abfrage von Bildern und Merkmalen, ein Forschungsbereich, der hier nicht näher betrachtet wird. Dieses Kapitel gibt einen Überblick über Systeme und Methoden der CBIR sowie relevante Arbeiten zur Klassifikation von Bildern.

Eines der ersten CBIR Systeme war QBIC (Query By Image Content) von IBM. QBIC verwendet Farbhistogramme, geometriebasierte Merkmalsextraktion und Texturbeschreibungen anhand von Kontrast. Ein weiteres bekanntes System ist BlobWorld, entwickelt an der University of California, Berkeley [Car+99]. BlobWorld

segmentierte Bilder mithilfe eines Algorithmus in zusammenhängende Regionen. Diese sogenannten Blobs basierten auf der Farbverteilung innerhalb der Regionen. Die daraus berechneten Farbhistogramme, dessen Funktionsweise bereits in Unterabschnitt 2.5.1 vorgestellt wurde, werden zu einem Histogrammvektor zusammengeführt. Die Distanz zwischen den Histogrammen ermöglichte die Suche nach ähnlichen Regionen in Bildern. Das SIMBA-System (Search Images By Appearance) von Siggelkow [SSB01], entwickelt in 2002, setzt auf einen globalen Ansatz. Dabei wird nicht wie bei BlobWorld ein Featurevektor aus den einzelnen Regionen erstellt, sondern direkt ein globales Histogramm aus den Farbkategorien. Der Fokus des Systems wurde bei der Entwicklung bereits, anders als bei BlobWorld, auf die Bildklassifikation gelegt. Fast funktionsgleich arbeitet das VIPER System von Squire und Müller [BI88] aus dem Jahr 1999. Es verwendet Farbhistogramme als Extraktionsmerkmal zur späteren Klassifikation anhand gespeicherter Merkmale in einer Datenbank. Dabei werden wie beim SIMBA System Farbhistogramme zur Erstellung eines Featurevektors extrahiert. Weitere erwähnenswerte Systeme mit ähnlichen Ansätzen sind das CIRES System von Iqbal und Aggrwal aus 2002 und das IRMA Projekt der Universität Aachen für die Entwicklung von Bildklassifikationsverfahren für radiologische Untersuchungen.

Die Literatur zeigt, dass Bildabrufsysteme den in Abschnitt 2.1 beschriebenen Phasen der **Bildvorverarbeitung**, **Merkmalsextraktion** und des **Bildvergleichs** folgen. Diese Architektur hat sich als bewährte Architektur in der Praxis und Forschung etabliert. Eine der größten Herausforderungen im Bereich der CBIR ist die Leistungsbewertung. Einheitliche Maßstäbe fehlen und qualitative Anforderungen an diese variieren je nach Forschungsgruppe. Abschnitt 3.4 greift das Problem in Bezug auf die Standardisierung der Distanzmetriken auf und stellt diesbezüglich relevante Literatur für den Bildabrufprozess vor. Ein weiteres Problem ist das Fehlen standardisierter Datenbanken für Vergleichszwecke. Autoren verwenden für Ihre vorgestellten Verfahren verschiedenste Datensätze mit unterschiedlichen Klassenhäufigkeiten, Bildqualität sowie Quantität. Um den Vergleich ihrer Forschungsergebnisse zu erleichtern, verwenden neuere Forschungsarbeiten standardisierte Bilddatensätze wie ImageNet, CIFAR-10 oder den Oxford RobotCar Datensatz. Trotz der voranschreitenden Standardisierung sind Ergebnisse, die auf unterschiedlichen Datensätzen basieren, derzeit nicht immer vergleichbar.

3.2 Vorverarbeitungstechniken

Wie in Abschnitt 2.3 dargelegt, ist es erforderlich, Bilder, die möglicherweise durch Störungen beeinträchtigt sind, zunächst in ein einheitliches Format zu überführen. In diesem Abschnitt werden relevante Arbeiten vorgestellt, die sich mit der Bildvorverarbeitung beschäftigen, um einen Ansatz für die vorliegende Arbeit zu entwickeln. Für eine so zu entwickelnde Klassifikationspipeline muss die Vorverarbeitung für jedes Verfahren dabei identisch gestaltet sein, um eine korrekte Vergleichbarkeit der Ergebnisse zu gewährleisten. Daher müssen Vorverarbeitungsmethoden adaptiv sein, um die weitere Verarbeitung mittels verschiedener Klassifikationsmethoden in der Nachverarbeitung zu gewährleisten.

3.2.1 Klassische Ansätze

In ihrem Paper „A Benchmark on Tricks for Large-scale Image Retrieval“ [Ko+19] untersuchen die Autoren die Effektivität verschiedener Vorverarbeitungsschritte für umfangreiche Datensätze (Oxford105k, 14.000 einzigartige Klassen und über 100.000 Bilder). Es wird demonstriert, dass Methoden wie Datensatzbereinigung und Augmentation die Performance einer Datenpipeline signifikant verbessern können. Die bereits in Abschnitt 2.3 erläuterten Vorverarbeitungsmethoden stellen gemäß [Yan+23] aktuelle Techniken für die Bildvorverarbeitung dar. Die entwickelte Pipeline in dieser Arbeit muss sich insbesondere mit der Frage befassen, wie diese Vorverarbeitungsschritte optimal kombiniert werden können, um aus Scans von durch Störungen beeinflussten Karten 2.3.1 diese Karten in einem einheitlichen Format für die weitere Verarbeitung zu extrahieren.

Ein interessanter Ansatz ist die in [MRT11] von den Autoren vorgestellte Pipeline für die automatische Erkennung von Pokerkarten. Diese nutzt den erstmals 1986 von John Canny eingeführten Canny Edge Detector [Can86a] zur Kantenerkennung. Zunächst wird das Eingabebild in Graustufen umgewandelt und anschließend mit einem Gaußschen Filter bearbeitet. Der Canny Edge Detektor identifiziert die Kanten im bearbeiteten Bild. Da Pokerkarten rechteckige Formen aufweisen, wird angenommen, dass alle inneren Winkel genau 90 Grad betragen müssen. Diese Logik ermöglicht es den Autoren, durch alle trapezförmige Konturen der erkannten Kanten zu iterieren und die zu behalten, welche inneren Winkel annähernd 90 Grad erreichen. Danach wird mittels perspektivischer Transformation das am besten passende Trapez aus dem Originalbild extrahiert. Die Autoren erläutern jedoch nicht, nach welchen Kriterien das optimale Trapez ausgewählt wird. Außerdem beinhaltet der immer monoton einfarbige Tisch, auf dem die Karte aufgenommen wird, keinerlei geometrische Strukturen. Dies erschwert die Anpassung des Verfahrens an andere Hintergrundstörungen, wie rechteckige Strukturen, die das Trapez verfahren nichtig machen würden.

3.2.2 Moderne Ansätze

Ein Ansatz für das Ausschließen des Hintergrundproblems wird in [Esc+18] vorgestellt. Die Autoren verwenden Instanz Segmentation, um innerhalb von Zellbildern jede Zelle einzeln zu klassifizieren. Für die erkannten Klassen werden Segmentationsmasken mit den jeweiligen Pixelbereichen einer Zelle generiert. Der Hintergrund der Zelle wird vom trainierten Netzwerk als eigene Klasse erkannt. Die Pixelgenauen Segmentationsmasken können daraufhin hinsichtlich Ihrer Ecken untersucht werden. Die jeweiligen Klassenobjekte wurden mit einer Genauigkeit von **98,1 %** und einem Recall von **99,6 %** identifiziert. Die Studie „A Survey on Instance Segmentation: State of the Art“ [HB20] vergleicht die aktuellsten Architekturen hinsichtlich ihrer Stärken und Schwächen, um einen umfassenden Überblick über den Stand der Technik zu bieten. Dabei werden auch die in [Esc+18] vorgestellten Objekterkennungs- und Segmentationsmethoden betrachtet. Die Architektur YOLO hat sich aufgrund ihrer einfachen Trainingsweise und ihrer generellen Genauigkeit bei der Segmentierung als effektiv erwiesen, jedoch ist die Trainingspipeline komplex und schwierig zu optimieren. Eine weitere Architektur, die 2022 vorgestellt wurde, ist RTMDet. Diese Architektur übertrifft die Yolo-Architektur in der Genauigkeit auf dem COCOval2017-Datensatz signifikant, insbesondere bei den Parametern von 10 Millionen bis 100 Millionen. Bei 62 Millionen Parametern wurde eine Genauigkeit von **51,5 %** erzielt. Mit der derzeit aktuellsten Architektur, Yolov11 [KH24], wurde diese Benchmark übertroffen. Bei mehr als 62 Millionen Parametern lag die

durchschnittliche Präzision der Bounding Box-Segmentierung bei **54,7 %** und stellt somit das leistungsfähigste Segmentierungsmodell auf dem cocoval Datensatz für die genannten Parametergrenzen zur Veröffentlichung dieser Arbeit dar.

3.3 Klassifikationsmethoden für Bildabrufsysteme

Dieser Abschnitt stellt verwandte Arbeiten in Bezug auf Klassifikationsmethoden vor und wägt diese anhand der Literatur hinsichtlich Ihrer Verwendbarkeit für Large Image Retrieval gegeneinander ab. Dessen Anforderungen wurden in Abschnitt 2.1 vorgestellt.

Im Forschungsbereich der Bildabrufsysteme liegt der Fokus angesichts der stetig wachsenden Leistungsfähigkeit von Convolutional Neural Networks (CNNs) in neueren Veröffentlichungen meist auf dem Training dieser Architekturen, um den Recall und die Genauigkeit des Bildabrufs zu optimieren. Die Forschung befasst sich einerseits mit auf Aufmerksamkeitsmechanismen basierenden Verfahren, wie Transformern [Dos+20; Li+23; Oqu+24; Car+21], und andererseits mit CNN-Architekturen [RTC19; TSJ15]. Diese modernen Klassifikationsmethoden sind für die hier zu entwickelnde Pipeline jedoch ungeeignet. Um gute Klassifikationsgenauigkeit mittels CNN basierten Verfahren zu erzielen, würde für jede einzigartige Klasse eine Vielzahl an Trainingsdaten erstellt werden müssen. Dies würde Trainingsdatensätze in der Größenordnung 100.000 - 1 Mio. Bilder erfordern, was wiederum eine fehlerfreie Augmentationspipeline voraussetzt und sehr ressourcenintensiv ist. Klassische Merkmalsextraktionsmethoden bieten in diesem Kontext eine praktikablere Alternative. Sie sind einfacher umzusetzen, da sie keine Trainingsdaten benötigen.

Ein gutes Beispiel für die Komplexität moderner CNN Architekturen ist die DELF Pipeline, die mit der Einführung der Deep Local Feature (DELF) Architektur in [MAJ13] eine Genauigkeit von **88,5 %** auf dem Oxford105k-Datensatz (102 einzigartige Klassen) erreichte. In ihrer Pipeline für die Bildabfrage nutzen sie Bilddeskriptoren als Merkmale und eine Baumstruktur für deren effiziente Speicherung und Abrufung. Das von ihnen vorgestellte System für großskalige Bildabfragen basiert auf der Erkennung lokaler Bildmerkmale, denen ein Aufmerksamkeitswert zugeteilt wird. Dementsprechend wird die Methodik der aufmerksamkeitsbasierten Einbettung, welches von der Transformer Architektur [Vas+17] vorgestellt wurde, hier mit dem in Prinzip des in ORB verwendeten Schlüsselpunktextraktion kombiniert. Die Autoren von DELF ließen sich von diesen Verfahren inspirieren und verwendeten den Ansatz des Tiefen neuronalen Netzes, welche zusammen mit der Aufmerksamkeitsmethodik der Transformer Architektur Regionen eines Bildes speichert, um es später mittels Distanzvergleich für den Bildabruf zu verwenden. Das daraus entstandene Modell bewertet die Bedeutung der Merkmale, um unnötige Informationen zu filtern und so den Abrufprozess zu verbessern.

Trotz seiner hohen Genauigkeit im Bildabrufprozess hat die Delf Pipeline die bereits erläuterten Nachteile von CNN Architekturen hinsichtlich Rechen- und Speicherintensität. Dementsprechend ist ein Blick auf weniger komplexe Merkmalextraktionsmethoden ratsam.

Die meiste Literatur im Kontext des Bildabrufs beschäftigt sich mit Datensätzen, die lediglich einige dutzend bis mehrere hundert einzigartige Klassen aufweisen. Ein für diese Arbeit wesentliches Problem ist dabei, dass

Klassifikationsmethoden bei zunehmender Menge einzigartiger Klassen an Genauigkeit abnehmen, wie später im Kapitel aufgezeigt wird. In ihrer Arbeit [Den+10] untersuchten die Autoren, wie verschiedene Klassifikationsmethoden mit 10.000 einzigartigen Kategorien umgehen können. Dabei wurden unterschiedliche Verfahrenskombination mittels SIFT Histogrammen als Featurevektoren verwendet und mittels Euklidischer Distanz und KNN Distanzberechnung mit einer Datenbank verglichen. Das beste Verfahren erzielte lediglich eine Klassifikationsgenauigkeit von **6,4 %** 2.6. Die Autoren von [SP11] griffen die Vorarbeit des vorhergehenden Papers auf und entwickelten einen auf Hash-Kernen und stochastischem Gradienten Abstieg basierenden Ansatz, der die Genauigkeit auf **16,4 %** erhöhte. Die verwendeten Architekturen der Forscher [SP11] und [Den+10] sind durch den Fokus auf Datensätze mit hoher Anzahl einzigartiger Klassen gering ausgefallen. Die Forscher lieferten jedoch wichtige Grundlagenforschung für die Analyse von Merkmalsextraktionsmethoden auf großen Datensätzen, indem sie zeigten, dass mit mehr als 10k einzigartigen Klassen unter Verwendung verschiedener Distanzmetriken ein zu geringer Präzisionswert erreicht wird. Experimente auf Datensätzen mit weit weniger einzigartigen Klassen sind in der Literatur seitdem gängiger. Im weiteren Verlaufe der Literaturrecherche wurden vor allem Paper mit Datensätzen gefunden, die zwischen wenigen dutzend bis mehreren hundert einzigartigen Kategorien aufweisen, was die erreichte Präzision der Verfahren beim Bildabruf sichtlich erhöht.

Ein erstes Beispiel ist das Paper [Xia+21] aus dem Jahr 2021 in welchem die Autoren eine erweiterte Version des Local Binary Pattern (LBP) vorstellen. Dieses ist speziell für den Einsatz im Bildabruf entwickelt. Die Forscher kombinierten adaptive Schwellenwerte mit richtungsabhängigen lokalen Binärmuster (LBP), um die Texturrichtungen zu repräsentieren. Das Verfahren baut demnach auf dem grundlegenden LBP Verfahren auf, welches in 2.5.2 bereits vorgestellt wurde. Dabei basiert die Berechnung des adaptiven Schwellenwerts auf dem Grauwert des zentralen Pixels und der Standardabweichung der Grauwerte in dessen Nachbarschaft. Die Forscher verglichen die vorgestellte Methode mit klassischen Methoden der Merkmalsextraktion, um eine Baseline vorzugeben. Die Experimente wurden auf dem Corel-1k-Datensatz durchgeführt. Die vorgeschlagene Methode erzielte eine Präzision von **67,63 %**, während die Vergleichsmethoden – darunter HOG-LBP, LF SIFT Histogramme, Farbhistogramme und LTP-moments – lediglich Ergebnisse zwischen **46 %** und **53 %** erreichten. Für die Ähnlichkeitsmessung der extrahierten Merkmale wurden die Euklidische- und Canberra-Distanzen verwendet, wobei Canberra die erwähnte höhere Präzision lieferte. Obwohl die Studie zeigt, dass die neue Methode eine höhere Präzision auf dem Corel-1k-Datensatz erreicht, ist die Aussagekraft für Anwendungen im Bereich des Large-Scale-Image-Retrievals begrenzt, da der verwendete Datensatz im Vergleich zu [Den+10] gering ausfällt.

Die im vorherigen Paper zum Vergleich herangezogene Methode mittels Histogram of oriented Gradients wird häufig im Image Retrieval verwendet. Die grundlegende Funktionsweise wurde in 2.5.1 vorgestellt. In der Veröffentlichung von [Hal13] wurde die HOG Methodik, sowie eine neue Variante mittels Bi-dimensional Empirical Mode Decomposition(BEMD) verwendet. Das Experiment nutzt den COIL-100-Datensatz, der 7.200 Farbbilder von 100 Objekten in 72 verschiedenen Orientierungen enthält. Dabei lieferte HOG als Referenzmethode bereits beachtliche Ergebnisse mit einer Präzision von **85,00 %** und einem Recall von **71,10 %**. Damit demonstriert das Paper zwar, dass HOG ein leistungsstarkes Werkzeug zur Extraktion lokaler Merkmale für kleinere Datensätze ist, gibt jedoch wie bereits im vorherigen paper keine Auskunft über die Verwendbarkeit für größere Datensätze.

Der Vorschlag der Autoren, HOG mit der BEMD-Technik zu kombinieren, verbesserte die Ergebnisse weiter. In der Kombination erreichen die beiden Techniken eine Präzision von **93,00 %** und einen Recall von **77,70 %**. Ein zentraler Kritikpunkt ist jedoch, dass die Autoren die Implementierungsdetails der BEMD-Technik nicht öffentlich zugänglich gemacht haben. Trotz des vorhandenen Pseudocodes erschwert dies eine Reproduktion innerhalb einer eigenen Implementierung. Für die Distanzberechnung zwischen den Merkmalen wurde ein Ansatz verwendet, der auf der Differenz der Durchschnittswerte der verwendeten Anzahl an Histogramm Kategorien basiert. Wenn auch nicht explizit von den Autoren genannt, ist dies die gleiche Vorgehensweise wie bei der Manhattan Distanz.

Eine weitere klassische Methode für die Merkmalsextraktion ist das Farbhistogramm, das in der Literatur häufig als Referenzmethode für die Einführung neuer Ansätze herangezogen wird. Im bereits diskutierten Paper [Xia+21] wurde die umfangreiche Studie der Autoren [DKN08] von der Universität Aachen zitiert, in der verschiedene Verfahren für Bildabrufsysteme untersucht und verglichen wurden. RGB-Histogramme, deren Funktionsweise bereits in Abschnitt 2.5.1 beschrieben wurden, erzielten mit einer durchschnittlichen Präzision von **58%** die besten Ergebnisse der Studie. Ein vergleichbares Ergebnis von **53,3%** wurden in der Studie mittels SIFT-Deskriptoren erreicht.

Die Studie führte die Verfahren auf fünf unterschiedlichen Datenbanken mit variierenden Bildmengen durch. Diese reichten von 1.000 Bildern mit 18 einzigartigen Kategorien bis zu 1.000 Klassen mit mehreren hundert Kategorien. Die Ergebnisse zeigen, dass Farbhistogramme für Bildabrufsysteme über verschiedene Datenbankgrößen hinweg konsistente und gute Leistungen erbringen können, erreichen allerdings nicht die Menge an einzigartigen Kategorien, die der Datensatz von [Den+10] enthalten hat. Die bereits erwähnte beste Durchschnittspräzision wurde dabei auf den vier Datensätzen mit jeweils lediglich 1.000 Bildern erreicht. Die ebenfalls enthaltene Datenbank IRMA 10k, die wesentlich größere Datenvielfalt bot, wurde nicht für den Vergleich der Farbhistogramme verwendet, was eine bessere Bewertung dessen Leistungsfähigkeit ermöglicht hätte.

Abseits der vorgestellten klassischen Merkmalsextraktoren beschäftigt sich die Forschung mit einem Ansatz, dessen Wurzeln in der Kryptografie liegt. Perceptual Hashing ist eine Methode zur Identifizierung und zum Vergleich von Bildern basierend auf ihrer wahrnehmbaren Ähnlichkeit. Es wird in der Praxis vor allem für den Schutz von Bildrechten angewandt. Traditionelle Ansätze zur Bildprüfung, wie etwa kryptografische Hash-Funktionen, sind empfindlich gegenüber Störungen, wie Rotation oder Kompression. Dies bedeutet, dass eine kleine Bearbeitung an einem Bild, etwa das Drehen oder Verkleinern, den Hash-Wert erheblich verändern und somit eine Identifizierung erschwert.

Im Gegensatz dazu hat sich Perceptual Hashing als robuster gegen bestimmte Manipulationen erwiesen, da es speziell von der Wahrnehmung des menschlichen Auges inspiriert ist. Forscher haben eine Vielzahl verschiedener Methoden für die Umsetzung von Perceptual Hashing vorgestellt [Zau10; DMY15; Tan+18; Qin+16a; Qin+16a; Rub+20]. Das Literaturreview State of the Art: Image Hashing [BM21], aus dem Jahr 2021, gibt einen guten Überblick über die Entwicklung der Forschung in diesem Bereich und dient daher weiterführend als Grundlagenliteratur.

Die referenzierten Verfahren, die auf RGB- und Gradientenmethoden basieren, zeichnen sich durch ihre Robustheit gegenüber Skalierungen und Rauscheffekten aus. Dies wurde bereits in Abschnitt 2.5.1 gezeigt. Jedoch bleiben Kompression und Rotationen im Eingabebild weiterhin problematisch.

Forscher untersuchten verschiedene Ansätze, um die Auswirkungen dieser Probleme zu negieren. Ziel der Forscher war vor allem eine Robustheit gegenüber Rotation zu erreichen. In der folgenden Auflistung werden die Ansätze diesbezüglich kurz zusammengefasst.

- **DCT-basiertes Hashing** [Zau10]: Skaliert das Bild auf 32×32 oder 16×16 Pixel, wendet die diskrete Kosinustransformation (DCT) an und extrahiert DCT-Koeffizienten zur Hasherstellung.
- **CSLBP** [DMY15]: Nutzt das zentral symmetrische lokale Binärmuster zur Darstellung von Bildinhalten.
- **Farbvektor-basiertes Hashing** [Tan+18]: Extrahiert und komprimiert das Histogramm aus Farbvektoren und generiert einen Hash aus diesem.
- **Strukturmerkmal-basiertes Hashing** [Qin+16a]: Führt Kanten-Erkennung durch, extrahiert DCT-Koeffizienten aus Blöcken und komprimiert diese zu einem Hash.
- **Ringpartitionen und Vektordistanz Hashing** [Tan+16b]: Verbessert Rotationsrobustheit durch extrahierte statistische Merkmale aus Bildringen im CIE $L^*a^*b^*$ Farbraum.
- **Farb / Gradienten Histogramm-basiertes Hashing** [Fan+17]: Generiert eine Hash-Feature-Map aus Farb- und Gradientenhistogrammen.

Die verwendeten Distanzmetriken und erzielten Genauigkeiten der referenzierten Verfahren werden in Tabelle 3.1 zum Vergleich aufgelistet. Es ist anzumerken, dass durch verschiedene Vorgehensweisen in den Experimenten einige Forscher keine durchschnittliche Präzision für Ihrer Retrieval Prozesse angegeben haben.

Es ist weitergehend erwähnenswert, dass aus dem Forschungsbereich des Perceptual Hashing durch das Aufkommen von modernen CNN's das sogenannte Deep Hashing entstanden ist [Xia+14; Jin+19]. Ein Deep Hashing Netzwerk bietet durch seine Resistenz gegen Rauschen und andere Störungsarten erhebliche Verbesserungen gegenüber dem klassischen Perceptual Hashing. Für das Training von Deep Hashing Netzwerken werden allerdings immense Datenmengen für das Training jeder Kartenkategorie benötigt. Dies macht die Methodik für kleine Datensätze unnötig komplex.

Ein weiterer interessanter Ansatz für die Bildklassifikation ist die Texterkennung. Die meiste Forschung im Bereich der Texterkennung beschäftigt sich mit Textextraktion für die Kategorisierung von Dokumenten. Die Ausnahme bildet das Paper der Autoren von: „Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm ", welches einen Ablauf speziell für Bildabrufsysteme vorstellen. Sie verwenden die auf Tesseract basierende Texterkennung, um Charaktere aus Bilddaten zu extrahieren und anschließend zu Wörtern zusammenzufügen. Dafür nutzen die Forscher unterschiedliche Auflösungen des Eingabebildes, um die Genauigkeit der Pipeline zu testen. Dabei stellte sich heraus, dass eine höhere Auflösung die Erkennungsrate der korrekten Wörter erhöhte. Den Eingabebildern wurden zuvor textuelle Label als Metadaten zugeordnet. Um einen Vergleich der extrahierten Wörter zu den Labeln zu ermöglichen, verwendeten die Forscher die Levenshtein-Distanz. Diese gibt, wie bereits in Abschnitt 2.5.3 vorgestellt, die Ähnlichkeit zweier Zeichenketten an als die Anzahl der Löschungen, Einfügungen oder Ersetzungen, die benötigt wird, um die eine Zeichenkette in die andere umzuwandeln. Nach Berechnung des ähnlichsten Bildes durch die Levenshtein-Distanz

anhand des beigefügten Labels kann das Bild aus der Datenbank abgerufen werden. Die Präzision variierte laut den Autoren je nachdem, wie hoch die Texterkennungspräzision war. Bei den hochauflösenden Bildern (2167 x 3064 Pixel) betrug die Genauigkeit bis zu 100 %.

Tabelle 3.1: Vergleich ausgewählter Merkmalsextraktionsmethoden

Referenz	Methode	Distanzmetrik (Bestes Ergebniss)	Genauigkeit (%)	DB: Anz. Kategorien
[Xia+21]	LBP	Canberra-Distanz	67,63	Corel1k: 10
[Hal13]	HOG	Manhattan	85,00	Coil-100: 100
[Hal13]	HOG + BEMD	Manhattan	93,00	Coil-100: 100
[DKN08]	RGB-Histogramm	Jensen-Shannon Divergenz	58,00	Viele: 18 - 100
[DKN08]	SIFT-Deskriptoren	Jensen-Shannon Divergenz	53,30	Viele: 18 - 100
[Den+10]	SIFT + SVM	Euclidian	6,40	ImageNet10K: 5225
[SP11]	Hash-Kerne	Punkt Produkt	16,40	ImageNet10K: 5225
[MAJ13]	DELF (Deep Local Feature)	Baumstruktur	88,50	Oxford105k: 102
[Zau10]	Luminanzbasiertes Hashing	Hamming	Keine Angabe	47
[DMY15]	CSLBP	Hamming	81,3	32
[Tan+18]	Farbvektorbasiertes Hashing	Hamming	> 90	1000
[Qin+16a]	Strukturmerkmals-basiertes Hashing	Hamming	> 98.6	200
[Tan+16b]	Ringpartitionen und Vektordistanz Hashing	Hamming	> 90	1338
[Fan+17]	Farb / HOG basiertes Hashing	Hamming	45,76	315
[AA21]	Texterkennung	Levenstein Distanz	100	14

Die in Tabelle 3.1 aufgelisteten Referenzen verwenden verschiedene Datenbankgrößen, Distanzmetriken und Pipeline Ansätze. Generell teilt sich die Literatur in Experimente auf Datensätze auf, die mehrere dutzend bis mehrere Tausend einzigartige Kategorien aufweisen. Die Forscher vermischen dabei Kombinationen aus Extraktionsmethoden und Distanzmetriken sowie Speicherlösungen, die für Ihre vorgestellten Methoden gut passen, ohne sich an Grundgerüsten zu orientieren. Die Referenzen verwenden allerdings alle die in 2.1 vorgestellten Phasen des Bildabrufs, was eine eigene Implementierung in diesem Vorbild sinnvoll macht. Bezüglich der analysierten Methoden ist zu erkennen, dass besonders Perceptual Hashing bei Kategorien mit mehr als 1000 einzigartigen Klassen Präzisionswerte $> 90\%$ erreicht. Aus der Literatur kann außerdem argumentiert werden, dass es derzeit keine Versuche gab, die vorgestellten Methoden auf einer einheitlichen Datenbank mit einer hohen Anzahl an Klassen und unterschiedlichen Distanzmetriken zu testen. Dies eröffnet einen interessanten Forschungsrahmen für einen eigenen Ansatz.

3.4 Distanzmetriken im Kontext des Bildabrufs

In Abschnitt 3.3 wurde bereits gezeigt, dass Forscher verschiedene Distanzmetriken zur Ähnlichkeitsberechnung in Ihren Arbeiten nutzen, ohne einem einheitlichen Standard zu folgen. Der Fokus des Kapitels lag auf Merkmalsextraktionsmethoden. Dieser Abschnitt fasst die in der Literatur beschriebenen Vergleiche von Distanzmetriken hinsichtlich ihrer Eignung für Bildabrufsysteme daher zusammen.

In einer Studie aus dem Jahr 2003, die mehr als 28.000 Bilder umfasste, wurden verschiedene Farb-Histogramme mit vier gängigen Distanzmetriken verglichen. Die Ergebnisse zeigen, dass die Manhattan-

Distanz die höchste Präzision bei der Bildabrufleistung erzielte [Sur+03]. Die Untersuchung ergab weiter, dass die Kosinus-Ähnlichkeit und die euklidische Distanz vergleichbare Leistungen erreichten.

Eine weitere Studie [KCB03] aus dem Jahr 2003 untersuchte die Effektivität verschiedener Distanzmetriken, speziell im Kontext von Gabor-Filtern als verwendete Extraktionsmethode. Insgesamt wurden neun Distanzmetriken evaluiert, darunter Manhattan, Euclidean, Chebyshev, Mahalanobis, Weighted-Mean-Variance, Canberra, Bray-Curtis, Squared Chord und Squared Chi-Squared. Die Bildmerkmale wurden mittels Gabor-Wavelets extrahiert und auf einer Datenbank mit 1856 Bildern getestet.

Die Studie zeigt, dass klassische Metriken wie Euclidean, Mahalanobis und Chebyshev in Bezug auf die Abrufgenauigkeit vergleichsweise schwach abschnitten (maximal 65,25 %, 57,00 % bzw. 52,75 %). Die Weighted-Mean-Variance Metrik erzielte mit 76,62 % eine deutlich bessere Leistung, durch die Normalisierung der Datenbank, welche Mittelwert und Varianz berücksichtigt. Besonders hervorzuheben sind die Canberra- und Bray-Curtis-Metriken, die mit einer Genauigkeit von 78,29 % die besten Ergebnisse erzielten. Diese Metriken minimieren negative Effekte der Skalierung, da ihre Distanzwerte stets kleiner als eins bleiben. Auch die Squared Chord- und Squared Chi-Squared-Metriken übertrafen die klassischen Ansätze, wenn auch mit geringeren Verbesserungen. Da die Gabor-Filter im Rahmen der Studie als Histogramme gespeichert wurden, können die Chord- und Chi-Square-Methoden als geeignete Ansätze für den Vergleich herangezogen werden.

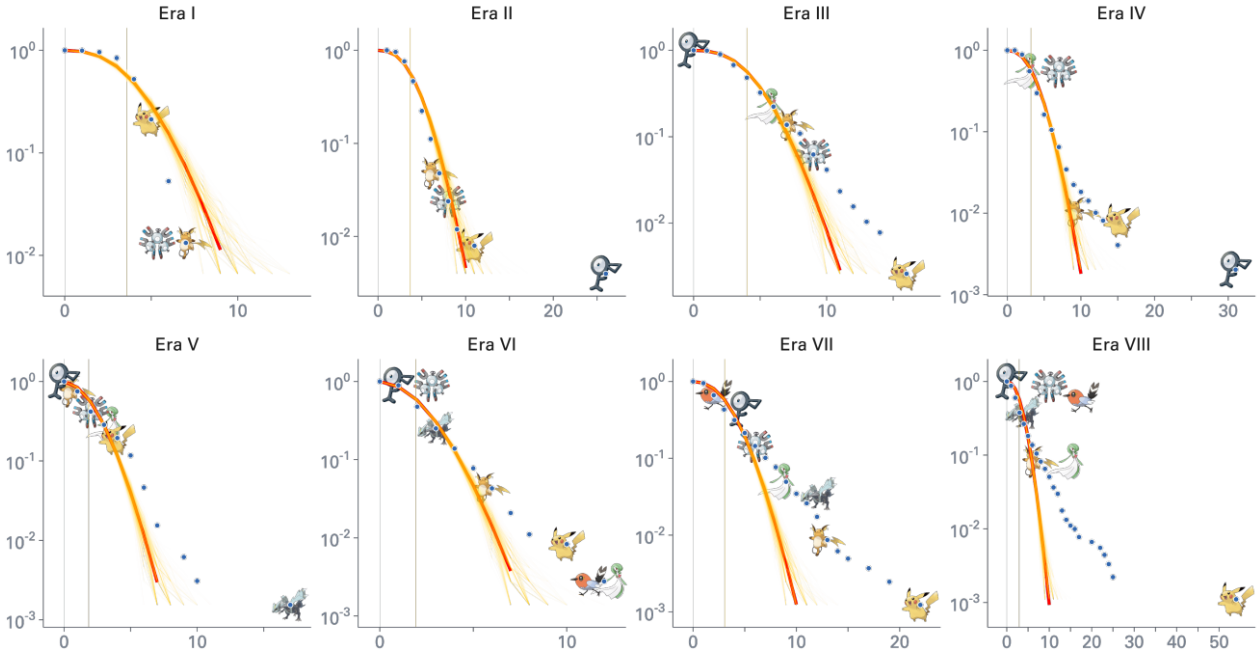
In einer weiteren Studie: „A Review on Image Retrieval Techniques"[MP16] wurden ebenfalls Distanzmetriken hinsichtlich ihrer Stärken und Schwächen beschrieben. Zu den betrachteten Metriken zählen Summe der absoluten Differenzen (SAD), euklidische Distanz, Manhattan-Distanz, Canberra-Distanz und Chebyshev-Distanz. Die Autoren fassten weiterhin die spezifischen Vor- und Nachteile dieser Metriken hinsichtlich Ihrer Verwendung für verschiedene Methoden auf. Während SAD besonders einfach berechenbar ist, erwähnen die Forscher einen Präzisionsabfall, wenn sich der Hintergrund eines Bildes ändert. Die euklidische Distanz wird als weit verbreitet und effizient beschrieben, erfordert jedoch die Berechnung der Quadratwurzelwert. Die Manhattan-Distanz ist aufgrund ihrer geringeren Rechenanforderungen beliebt, kann jedoch bei sehr ähnlichen Bildern zu falschen Ergebnissen führen. Die Canberra-Distanz bietet eine Normalisierung der Differenzen, ist dabei jedoch nicht robust gegenüber Ausreißern in der Häufigkeitsverteilung. Die Chebyshev-Distanz berücksichtigt den größten Unterschied in den Merkmalen, was in bestimmten Fällen von Vorteil sein kann, jedoch kleinere Unterschiede vernachlässigt.

Wie aus der Literatur hervorgeht, ist die Wahl der richtigen Distanzmetrik entscheidend für die Leistung eines Bildabrufsystems. Sie beeinflusst die Präzision bei der Ähnlichkeitsberechnungen und kann bei richtiger Wahl die Robustheit gegen Störungen im Bild erhöhen. Verschiedene Metriken bieten unterschiedliche Vorteile, die je nach Merkmalsarten und Anwendungsfall variieren. Die Entwicklung eines einheitlichen Ähnlichkeitsvergleichs für verschiedene Merkmalsextraktionsmethoden erscheint daher ratsam, insbesondere da die gefundene Literatur keine Auskunft über die Leistungsfähigkeit hinsichtlich der in Abschnitt 3.3 vorgestellten Verfahren bietet.

3.5 Datenlage und statistische Herausforderungen

Der folgende Abschnitt geht auf Literatur bezüglich des Pokémon SKS ein, um die statistische Distribution der enthaltenen Karten zu analysieren.

Die Autoren von [Mat23] erheben in Ihrer Arbeit die Verteilung von Pokémon-Karten im genannten SKS. Sie zeigt, dass Designprozesse oft fälschlich als zufällig betrachtet werden, obwohl Verzerrungen wie positive Frequenzabhängigkeit (z.B. die Beliebtheit von Pikachu) eine Rolle spielen. Die Autoren entwickelten ein neutrales Modell, bei dem Karten zufällig aus einem Pool generiert werden. Echte Daten zu 12.325 Karten des Pokémon SKS (1999–2022) zeigen jedoch Abweichungen: Besonders ab der dritten Serie, siehe Abbildung 3.1, tritt eine „Long-Tail"-Verteilung auf, bei der einige Pokémon häufiger erscheinen als erwartet. Pikachu bleibt durchgehend populär. Die Autoren vermuten daher eine Verzerrung im Designprozess.



Die x-Achse zeigt die Anzahl der Pokémon-Varianten, während die y-Achse die logarithmierte Wahrscheinlichkeit darstellt, ein Pokémon mit dieser Häufigkeit zu finden.

Abbildung 3.1: Verteilung von Pokémon in den Versionen [Mat23]

Um mögliche Verzerrungen zu erkennen, untersuchten die Forscher, ob Designer neue (pro-Neuheitsbias) oder klassische Pokémon (pro-klassischer Bias) bevorzugen. Des Weiteren wird untersucht, ob jedes Pokémon mindestens einmal abgebildet wird (anti-Abwesenheitsbias). Ein pro-Neuheitsbias führt dazu, dass neue Pokémon häufiger als Kartenvarianten erscheinen, während ein pro-klassischer Bias etablierte Pokémon begünstigt. Klassifikationsmodelle, die diese Präferenzen ignorieren, könnten die Häufigkeit bestimmter Pokémon falsch vorhersagen. Der Anti-Abwesenheitsbias zeigt laut der Autoren, dass Designer bemüht sind, alle Pokémon mindestens einmal zu integrieren. Modelle, die dies nicht berücksichtigen, könnten fälschlicherweise annehmen, dass bestimmte Pokémon dauerhaft ausgeschlossen sind. Zudem verstärkt der „Rich-Get-Richer"-Effekt, dass beliebte Pokémon immer häufiger vorkommen. Modelle, die diese Verzerrungen nicht einbezie-

hen, liefern ungenaue Ergebnisse. Die Verzerrungsthesen der Autoren sind insbesondere für die spätere Evaluation eines zu entwickelnden Testszenario beachtenswert, da einige Testbilder eine höhere Häufigkeit in der Bilddatenbank aufweisen könnten und dementsprechend die pro Klassen Genauigkeit bei der Evaluation verfälschen [Mat23].

Kapitel 4

Konzeptionierung

Das folgende Kapitel beschreibt die Konzeption für die zu entwickelnde Datenpipeline für die Vorverarbeitung und Klassifikation von Pokémon Karten. Zunächst werden in Kapitel 4.1 die Anforderungen an die Pipeline, welche auf den in Abschnitt 3 gewonnenen Erkenntnissen basieren, aufgestellt und die daraus entstandene Forschungsfrage vorgestellt. Darauf folgt in Abschnitt 4.2 eine Beschreibung des experimentellen Aufbaus, welcher die wissenschaftliche Grundlage für das weitere Vorgehen schafft und sich dabei an einem Grundgerüst orientiert. Auf Basis des in Abschnitt 4.2 vorgestellten Grundgerüsts werden die in Abschnitt 4.3, 4.4.1 sowie 4.4.2 folgend vorgestellten Experimente konzeptioniert und anschließend das in Abschnitt 4.5 beschriebene Test- und Validierungskonzept für deren Auswertung erstellt. Dieses Kapitel beschäftigt sich dabei lediglich mit den betrachteten Problemen der Arbeit und den theoretischen Konzeptionen, die deren Lösung dienen. Die genaue Umsetzung der hier vorgestellten Konzepte wird im Implementierungskapitel 5 beschrieben.

4.1 Anforderungen

Aus der Literatur ergeben sich folgende Anforderungen an die Entwicklung eines eigenen Bildabrufsystems:

1. **Standardisierte Vorverarbeitung:** Die Literatur hat gezeigt, dass eine einheitliche und gut funktionierende Vorverarbeitung maßgeblichen Einfluss auf die Qualität der Leistung von Bildabrufsystemen hat. Dementsprechend muss ein eigener Ansatz den Fokus zunächst auf die Optimierung dieses Prozesses setzten.
2. **Vergleich von Extraktionsmethoden:** In den in Tabelle 3.1 gefundenen Merkmalsextraktionsmethoden basieren teilweise auf Studien, die verschiedene Methoden kombinieren, allerdings ohne Ihre Forschungsthematik zu überschneiden. Beispielsweise fehlt ein Vergleich der eher klassischen Methoden, mit Perceptual-Hashing, sowie der Texterkennung. Der zu entwickelnde Ansatz sollte eine Kombination von den in der Literatur gefundenen Extraktionsmethoden untersuchen, um einen einheitlichen Vergleich zu ermöglichen.

3. **Vergleich von Distanzmetriken:** Vergleichbar mit dem vorherigen Argument zu Merkmalsextraktionsmethoden existiert in der Literatur auch kein umfassender Vergleich verschiedener Distanzmetriken in Bezug auf die Ähnlichkeitsberechnung von Bildern. Wo sinnvoll, sollten verschiedene Distanzmetriken verglichen werden, um deren Einfluss auf die Genauigkeit zu untersuchen.

4. **Einheitliche Datenbank:** Die Literatur verwendet verschiedenste Datenbanken für die Bewertung Ihrer vorgestellten Methoden. Dies zeigt auf, dass für die Entwicklung eines eigenen Konzepts eine Datenbank entworfen werden muss, die eine hohe Anzahl an einzigartigen Klassen (Pokémon Karten) aufweist. Diese Einheitlichkeit des Datensatzes ermöglicht einen Vergleich der bereits angeführten Extraktionsmethoden und Distanzmetriken und kann für darauf aufbauende Forschung eine modifizierbare Datenbasis bieten.

Aus diesen Anforderungen lässt sich weitergehend die folgende Forschungsfrage ableiten:

„**Welche Kombination aus Bildvorverarbeitungsmethoden und Klassifikationsalgorithmen innerhalb einer Datenpipeline ermöglicht die genaueste eindeutige Klassifizierung von Pokémon Sammelkarten unter verschiedenen Störeinflüssen auf das Eingabebild.**“

Das aus der Forschungsfrage definierte Ziel ist es demnach, eine Datenpipeline für die Kartenklassifikation zu erstellen, um die Forschungsfrage zu beantworten.

4.2 Experiment

Um eine methodisch fundierte Evaluation der vorgestellten Forschungsfrage zu gewährleisten, wird ein experimenteller Rahmen entwickelt. Die Untersuchung zielt darauf ab, die Genauigkeit der Methoden zur Klassifikation von Sammelkarten zu untersuchen, indem die Datenpipeline in zwei Schlüsselbereichen modifiziert wird: der Bildvorverarbeitung und der Merkmalsextraktion. Die übergreifende Struktur der Pipeline, an der sich die durchzuführenden Modifikationen orientieren, wird anhand des Grundgerüsts in Abbildung 4.1 vorgestellt.

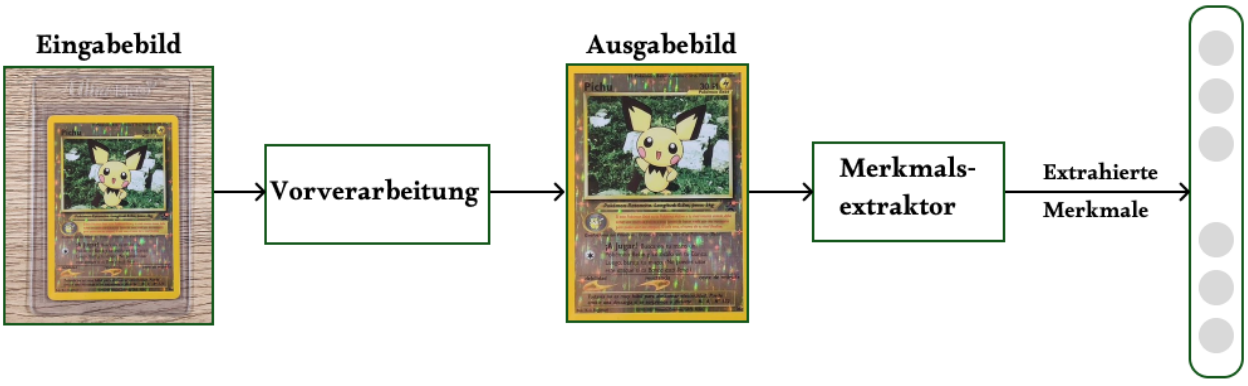


Abbildung 4.1: Grundgerüst der Datenpipeline

Der Vorverarbeitungsschritt hat zum Ziel, die Karte zunächst pixelgenau aus dem Umgebungshintergrund zu segmentieren. Die erkannte Karte kann dabei rotiert und oder gekippt aufgenommen worden sein. Durch die Anwendung einer perspektivischen Transformation wird ein Eingabebild in die standardisierte Vogelperspektive überführt. Der detaillierte Prozess mit den durchzuführenden Modifikationen am Grundgerüst wird in Abschnitt 4.3 vorgestellt. Das entwickelte Vorverarbeitungskonzept wird hinsichtlich seiner Erkennungsgenauigkeit ausgewertet und evaluiert. Der Vorverarbeitungsprozess ist demnach Grundlage für weitere Durchführungen des Experiments und bietet einen einheitlichen Ansatz für die Durchführung, der nachfolgenden Experiment hinsichtlich der Merkmalsextraktion.

Die restlichen in Abschnitt 4.1 vorgestellten Anforderungen für die Standardisierung der Extraktionsmethoden, Distanzmetriken und einheitlichen Datenbank wird im Test- und Validierungskonzepts in Abschnitt 4.5 des weiteren Verlaufs dieses Kapitels vorgestellt. Nach der Vorverarbeitung wird ein Merkmalsextraktionskonzept entwickelt, um die Klassifikationsgenauigkeit der verschiedenen Methoden zu vergleichen. Die spezifischen Methoden der Merkmalsextraktion sind im Abschnitt 4.4 ausführlich beschrieben. Nach der Vorverarbeitung wird ein Merkmalsextraktionskonzept entwickelt, um die Klassifikationsgenauigkeit bei Kombination verschiedener Methoden miteinander zu vergleichen. Die dafür verwendeten Distanzmetriken und spezifische Methoden der Merkmalsextraktion sind in Abschnitt 4.4 ausführlich beschrieben. Jedes Szenario wird mit einer umfassenden Analyse der Klassifikationsgenauigkeit abgeschlossen, wobei die Wirksamkeit der Vorverarbeitung und Merkmalsextraktion in Bezug auf die Gesamtleistung der Pipeline bewertet wird und anhand des Testszenarios aus Abschnitt 4.5 ausgewertet.

4.3 Vorverarbeitungsstrategie

Dieser Abschnitt behandelt die Entwicklung eines eigenen Konzeptes für die Vorverarbeitung der Sammelkarten für die spätere Klassifikation.

In einem ersten Schritt sollen eine oder mehrere Sammelkarten, die innerhalb einer Fotografie vorliegen, eindeutig erkannt werden. Für die dafür benötigte Objekterkennung und Instanzierung wird das Aktuellste in der Literatur gefundene Yolo Instanzsegmentationsmodell verwendet.

Konturdetektion
Zunächst wird ein vortrainiertes YOLO-Modell auf eigens entwickelte Trainingsdaten fein abgestimmt. Die Umsetzung des Trainings für diese Arbeit wird in Abschnitt 4.3.1 detailliert erläutert. Ein so trainiertes Modell bietet die Möglichkeit, Eingabebilder hinsichtlich Ihrer enthaltenen Sammelkarten zu untersuchen, indem das Modell alle Konturbereiche berechnet, die laut trainierten Modell die höchste Wahrscheinlichkeit haben der Klasse anzugehören. Diese Klassenzugehörigkeitsberechnung durch Konfidenzbewertung wurde bereits im Theorieteil in Abschnitt 2.4.3 vorgestellt. Diese sogenannten Segmentationsmasken kennzeichnen demnach alle Bereiche, in denen Sammelkarten im Bild vorhanden sein können und sind Grundlage für die Segmentation vom Hintergrund. Dieser erste Schritt der Vorverarbeitung wird in Abbildung 4.2a dargestellt.

Konturapproximation

Die erkannten Konturen, die das Segmentationsmodell als Ausgabe bereitstellt, können Verzerrungen und Ausreißer in den Datenpunkten besitzen. Beispielsweise könnten Pixel des Hintergrundes falsch als Kartenklasse identifiziert worden sein. Um die Karten aus den Konturen zu extrahieren ist es daher ratsam an Stelle der Segmentationsmasken eine vereinfachte Geometrieform zu erzeugen, die eine einheitliche Ausgabe sicherstellt. Die einfachste und für den Zweck von Sammelkarten geeignetste Form ist eine Rechteckapproximation. Sie ermöglicht, die Kontur des Sammelkartenobjektes durch vier Eckpunkte zu beschreiben, was die Extraktion der Karte im nächsten Schritt vereinfacht. Dafür wird zunächst der Umfang der Kontur berechnet und mittels einer Vereinfachung die Eckpunkte der Kontur auf vier reduziert. Da Pokémon Sammelkarten in der Regel rechteckig sind und vier gerade Kanten besitzen, können wir nur diejenigen Konturen weiterverarbeiten, die genau vier Kanten aufweisen. Diese Annahme ermöglicht es uns, andere Objekte, die möglicherweise mehr oder weniger als vier Kanten haben, auszuschließen und uns auf die eigentliche Karte zu konzentrieren. Die genaue Umsetzung des Verfahrens wird im Implementierungsabschnitt 5.5 erläutert. Die so approximierte Form einer Kontur wird in Abbildung 4.2b dargestellt.

Eckpunkterkennung

Im vorletzten Schritt müssen die bereits approximierten Ecken der Form für den nächsten Schritt noch richtig sortiert werden. Diese Eckpunkterkennung und Zuordnung dient der anschließenden perspektivischen Transformation Diese Phase der Eckenkennzeichnung wird 4.2c abgebildet.

Perspektivische Transformation

Abschließend erfolgt die perspektivische Transformation. Eine eventuell rotierte oder durch den Kamerawinkel gekippte Sammelkarte soll so bearbeitet wird, dass es aus der Vogelperspektive und in gerader Ausrichtung erscheint. Für diesen Zweck wird das in Abschnitt 2.3.4 vorgestellte Verfahren der perspektivischen Transformation angewandt. Durch die Projektion der Ebene der erkannten Form auf die erwünschte Ebene der Vogelperspektive wird die Sammelkarte in ein einheitliches Format gebracht. Der dafür verwendete Code wird in Abschnitt 5.5 vorgestellt. Diese letzte Phase der perspektivischen Transformation ist Abbildung 4.2d abgebildet.

(a) Gefundene Kontur (b) Approximierte Form (c) Kenzeichnung der Ecken (d) Transformation der Perspektive

Abbildung 4.2: Schritte des Vorverarbeitungskonzepts

Durch diese vier Phasen wird die Karte pixelgenau isoliert und auf die gleiche Ausrichtung und Perspektive

wie die originalen Bilddaten gebracht. Dies ermöglicht einen einheitlichen Vergleich der im weitere Verlaufe dieses Kapitels vorgestellten Merkmalsextraktionsmethoden.

4.3.1 Feinabstimung des Segmentationsmodels

Das Yolov11 Segmentationsmodell ist bereits in der Lage durch sein Vortraining viele Objektklassen von Werk aus mit einer hohen Genauigkeit zu erkennen. Für die spezifische Erkennung von Sammelkarten fehlt allerdings diese Fähigkeit. Daher ist eine Feinabstimmung notwendig, um das Modell auf die Erkennung der besonderen Form der Karte abzustimmen. Im Folgenden erfolgt die Vorstellung des Konzepts der Feinabstimmung, während die genaue Umsetzung in Abschnitt 5 des Implementierungskapitels vorgestellt wird.

Um eine gute Generalisierung des Modells auf verschiedene mögliche Bildhintergründe zu erreichen, ist es nötig, die für das Training zur Verfügung stehenden Bilddaten zu augmentieren. Dazu werden Bilddaten automatisiert augmentiert, um eine größere Vielfalt an Trainingsbildern zu erzeugen. In Abbildung 4.3 ist die Pipeline für die Erstellung des Segmentationsmodels visualisiert. Das Ziel der Augmentierung ist es, das Modell auf unterschiedliche Hintergründe vorzubereiten, wie beispielsweise Tische, Böden, Teppiche und andere Umgebungen, in denen die Sammelkarten auftreten können.

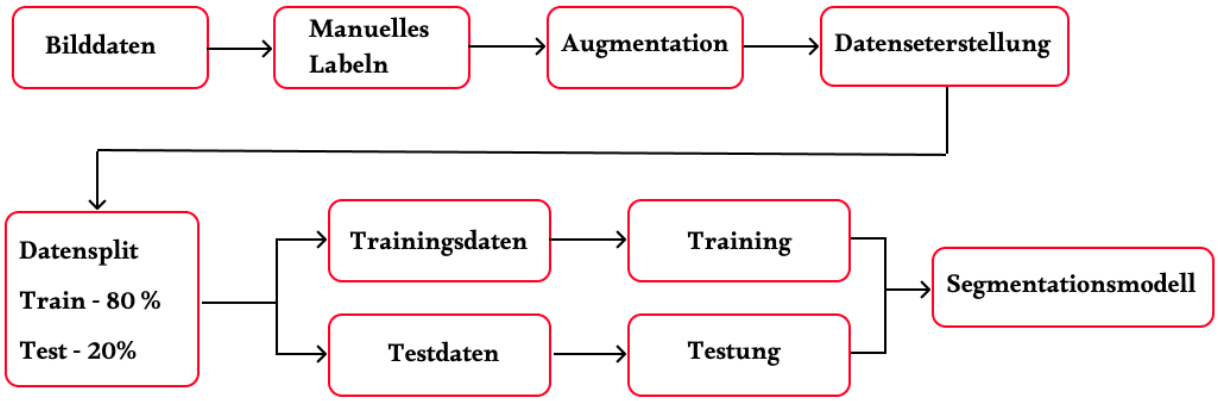


Abbildung 4.3: Pipeline für die Erstellung des Segmentationsmodell

Der so generalisierte Datensatz wird in Trainings- und Testbilder Bilder geteilt. Die Trainingsdaten werden für die Feinabstimmung des Models verwendet, während die Testdaten in jeder Trainingsiteration für die Berechnung der Klassifikationsgenauigkeit der erzeugten Segmentierungen verwendet wird. Die Evaluation des trainierten YOLO-Modells erfolgt mithilfe eines separaten Testdatensatzes, der 15% des gesamten Datensatzes ausmacht. Dieser Testdatensatz wird verwendet, um die Leistung des Modells auf neuen, bisher ungesehenen Daten zu überprüfen. Während des Evaluationsprozesses wird die Erkennungsgenauigkeit des Modells anhand der in Abschnitt 2.6 vorgestellten Evaluationskriterien für Segmentationsmodelle ermittelt. Das so erzeugte und evaluierte Modell wird für die weitere Verwendung in der Datenpipeline herangezogen.

4.4 Strategie zur Merkmalsextraktion

In diesem Kapitel werden die ausgewählten Verfahren der Merkmalsextraktion aus der Literaturrecherche hinsichtlich Ihrer Verwendbarkeit für den zu entwickelnden Prototypen betrachtet. Daraufhin erfolgt eine Entscheidung, ob diese in die Implementierung aufgenommen werden.

4.4.1 Histogrammbasierte Verfahren

Aus der Literaturrecherche ging hervor, dass besonders Histogramm basiert Verfahren für die Merkmalsextraktion herangezogen werden. Für die Überprüfung deren Nutzungsfähigkeit hinsichtlich der Klassifikation von Pokémon Karten müssen zunächst einige Grundlasten der Technik abgewägt werden. Der wichtigste Parameter bei der Konzeption eines Histogrammverfahrens ist die Wahl der Anzahl der Bins innerhalb eines Kanals für die jeweiligen extrahierten Merkmale einer Methode. Für das RGB Verfahren wären beispielsweise die Farbkanäle Rot, Grün, Blau gegeben und die Anzahl der Bins entscheidet über die Feingranularität der Farberfassung.

Die Anzahl der Bins hat direkten Einfluss auf die Klassifikationsgenauigkeit, da mehr Kanäle die Merkmalsunterschiede feiner auflösen, jedoch gleichzeitig dazu führen kann, dass unwichtige oder rauschbedingte Merkmale zu stark gewichtet werden. Außerdem entscheidend für die Konzeption eines Klassifikationsverfahrens ist die Wahl von Verfahren, die invariant gegenüber den Störungen eines Kartenscans sind. Da wir in den Anforderungen von Bildern mit unterschiedlichen Lichtverhältnissen ausgehen, ist der RGB Farbraum für die Extraktions ungeeignet, da Änderungen in der Helligkeit sich direkt auf alle drei Kanäle auswirken.

Im HSV-Farbraum wird die Helligkeit im separaten value Kanal gespeichert und beeinflusst somit nicht die Sättigung und den Farbton.
Andere farbunabhängige Methoden, die in der Literatur mit guten Klassifikationsgenauigkeiten identifiziert worden sind, HOG und LBP. Wie in 2.5.1 bereits vorgestellt ist HOG darauf ausgelegt, die Häufigkeit lokalen Kantenrichtungen in einem Bild zu erfassen, indem die Richtungsgradienten berechnet werden. Da Kantenrichtungen auf Unterschieden benachbarter Pixel basieren und nicht auf absoluten Farbwerten, ist das Verfahren invariant gegenüber Änderungen der Beleuchtung.
LBP wiederum, wie in 2.5.2 vorgestellt, misst relative Intensitätsänderungen innerhalb lokaler Bildbereiche. Dies macht es ebenfalls invariant gegenüber Lichtverhältnissen, hat allerdings durch den Fokus auf Pixelintensität eine Empfindlichkeit gegenüber Bildrauschen.

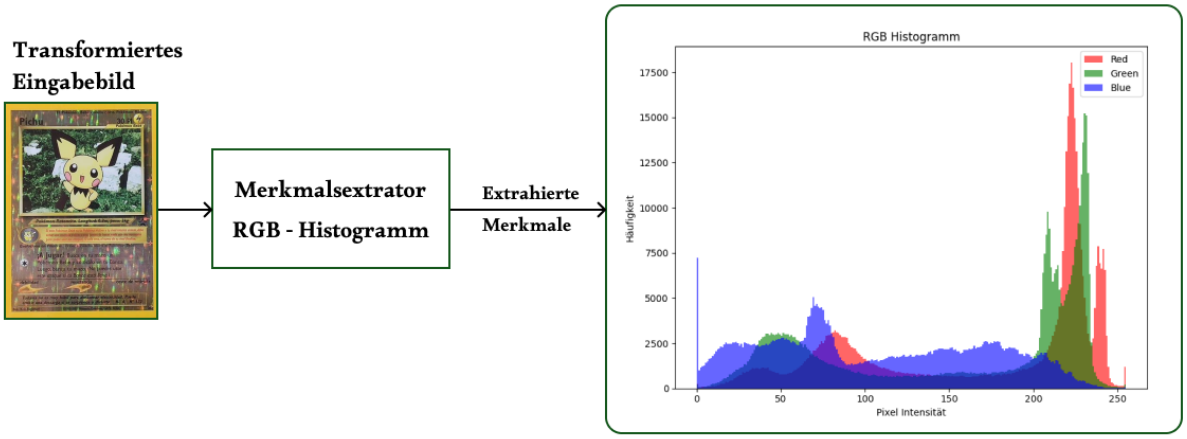


Abbildung 4.4: Merkmalsextraktion mittels RGB Histogramms

Demnach eignen sich die drei Verfahren der LBP, HOG, HSV Histogrammextraktion für den hier entwickelten Ähnlichkeitsvergleich.

Um die vorgestellte Problematik der hohen Dimensionalität zu reduzieren, kommen zweierlei Verfahren infrage. Entweder wird die Dimensionalität durch ein Verfahren wie das Principal Component Analysis in einem oder mehreren der gewählten Kanäle reduziert, oder das Eingabebild wird insoweit verkleinert, als die Dimensionalität sich stark reduziert. Da für die Vorverarbeitungsphase bereits ein Yolo Instanzsegmentationsmodell entwickelt wurde ergibt sich der Ansatz einer Feinanpassung des Modells, welcher nur das Artwork der Karten extrahiert. Dieser Ansatz ist jedoch Problematisch, da die Artworks sich je nach Kartenversion über die gesamte Karte erstrecken, und bei einigen Karten spezielle Symbole und andere Störeffekte das Artwork verdecken. Ein darauf angepasstes Segmentationsmodell würde die erkannten Artworks dementsprechend falsch segmentieren und das Ergebnis verfälschen. Ein weiterer Ansatz für die Reduzierung der Dimensionalität ist die Anwendung des in Abschnitt 2.3.3 vorgestellten DCT Verfahrens, welches die aussagekräftigsten Merkmale aus den extrahierten Histogrammen zusammenfasst. Dies verringert die Dimensionalität der extrahierten Merkmale.

Die Extraktion eines Featurevektors wird in Abbildung 4.4 anhand der RGB Methode veranschaulicht. Die Histogrammbasierten Methoden, können durch die in Abschnitt 2.5.3 eingeführte Chi-quadrat-Distanz hinsichtlich ihrer Klassifikationsgenauigkeit ausgewertet werden.

4.4.2 Hashing Verfahren

Methoden des Perceptual Hashings haben sich in der Literatur durch ihre besonders hohe Klassifikationsgenauigkeit bei Datensätzen mit mehr als 1000 einzigartigen Kategorien hervorgehoben. Von den vorgestellten Verfahren in Tabelle 3.1 schnitten CSLBP, Farbvektorbasiertes Hashing, Strukturmerkmals basiertes Hashing und das Ringpartition und Vektordistanzhashing mit Genauigkeiten von 81.3 bis 98.6 % besonders gut ab. Lediglich das CSLBP Verfahren ist nicht für die hier vorgesehene Merkmalsextraktion geeignet, da die Autoren

des Algorithmus die Implementation speziell für kryptografische Schlüsselsicherheit entwickelt haben. Die Implementation anhand des Papers entfällt daher. Der von [Zau10] vorgestellte DCT basierte Perceptual Hash hat zwar keine direkte Klassifikationsgenauigkeit im Paper angegeben, hat in der Literatur allerdings viele andere Arbeiten und Algorithmen inspiriert. Eine Aufnahme in die Pipeline ist daher angebracht für einen umfangreichen Vergleich.

Die restlichen Methoden werden als Methoden der Merkmalsextraktion in der Pipeline umgesetzt, um sie hinsichtlich Ihrer Einsetzbarkeit für Pokémon Karten zu untersuchen. Je nach Methode wird entweder ein Hash im binären oder Dezimalformat ausgegeben. Für den Vergleich von binären Ausgaben eignet sich die Hamming-Distanz. Für den Vergleich von Vektoren basierend auf Dezimalwerten kann die Euklidische Distanz und die Cosinus Vektor Distanz herangezogen werden.

Alle genannten Methoden wurden in Abschnitt 2.5.3 vorgestellt.

Bei der betrachteten Anzahl an einzigartigen Pokémon Karten ist wichtig eine Bitlänge zu wählen, die Kollisionen bei Erzeugung des Hash verhindert. Die Wahl dieser Länge ist für die Methoden zu betrachten, welche einen binären Ausgaberaum voraussetzen. Um den Einfluss dieser Störung auf die Klassifikation zu negieren wird der 128 Bit Ausgaberaum als Bitlänge gewählt. Bei 16 Byte Hash-Länge sind $2^{128} \approx 3.4 \times 10^{38}$ Signaturen möglich. Die Kollisionswahrscheinlichkeit ist hier somit negierbar. Ein längerer Ausgaberaum führt bei den Hashverfahren zu genaueren Ergebnissen, da die Hash-Länge direkten Einfluss auf die Anzahl an erfassten Bildmerkmalen hat. Je länger der Hash, desto granularer können Bilder unterschieden werden.

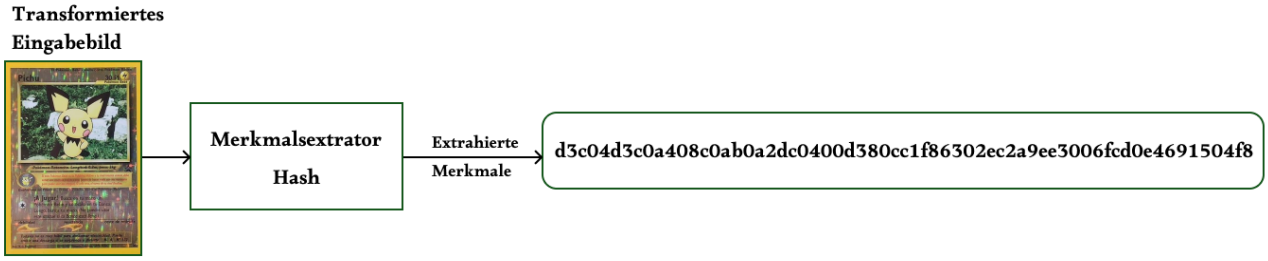


Abbildung 4.5: Merkmalsextraktion mittels Perceptual Hash

Für die Hash basierten Verfahren wird anders als bei den Histogrammbasierten Verfahren die unmodifizierte Vorverarbeitungspipeline, wie in 4.3 vorgestellt, verwendet. So wird das gesamte vorverarbeitete Ausgabebild an die Verfahren übergeben. Die Implementation der Verfahren wird detailliert in Abschnitt 5, wo vorhanden, nach Vorgabe aus den referenzierten Papern erfasst. Die Extraktion anhand des Perceptual Hash wird in Abbildung 4.5 veranschaulicht. Wenn in der Literatur keine Instruktion für die Algorithmen gegeben ist, wird dieser selbst implementiert und vorgestellt.

4.4.3 Texterkennungsbasiertes Verfahren

Das in Abbildung 2.2 vorgestellte Layout von Pokemon Sammelkarten ermöglicht es, die Karten hinsichtlich Ihrer einzigartigen Textmerkmale zu identifizieren. Um eine Einmaligkeit des extrahierten Vektors zu garantieren, muss der Name des Pokémon, sowie die Setnummer gefunden werden. Dafür wird eine Werkzeugkette

benötigt, die ausgewählte Bereiche der Karte nach den beiden Merkmalen durchsucht und extrahierten Text hinsichtlich seiner Ähnlichkeit zu den in der Datenbank gespeicherten Formaten abgleicht. In der gefundenen Liteartur zur Textverarbeitung mittels Optical Character Recognition nutzen die Forscher die Tesseract Engine, welche von Google entwickelt wurde. Da diese frei erhältlich und lizenzfrei verwendbar ist, wird die Engine für das folgende Konzept verwendet. Die Kartenlokalitäten des Namens und der Kartensetnummer haben sich im Laufe der Zeit immer wieder verändert. Die generelle Positionierung blieb jedoch gleich. Der Name befindet sich im oberen Bereich der Karten, während die ID entweder auf der linken oder rechten Seite des unteren Bildbereichs lokalisiert wird. Nach durchgeführter Vorverarbeitung liegt eine Karte in der Vogelperspektive vor. Die Karte wird nun partitioniert, um lediglich den untern und den oberen Bereich der Karte für die Texterkennung zu extrahieren und so die extrahierte Textmenge zu verkleinern. Die daraus partitionierten Kartenanteile werden in Abbildung 4.6 visualisiert.

Abbildung 4.6: Merkmalsextraktion mittels Kartenpartitionierung

Aus der Literatur ging hervor, dass Nachbarschaftsfenster verschiedener Größen Einflüsse auf die Texterkennungsqualität hat. Dies wird in der Implementierung in Abschnitt 5.6.3 berücksichtigt. Im ersten Schritt wird nun mittels Regular Expression der extrahierte Text auf das Format der ID hin untersucht. Die Setnummer, welches folgendermaßen aussehen kann: `36/108`, wird für das Regex Pattern folgend definiert:

Definition: Eine oder zwei Ziffern, gefolgt von einem Schrägstrich, und anschließend zwei oder drei Ziffern:

```
\b\d{1,2}/\d{2,3}\b
```

Wenn eine ID durch die Regeltechnik erkannt wurde, werden im nächsten Schritt alle Karten aus der Datenbank geladen, die der gleichen Setnummer zugeordnet sind. Nun wird der obere Bereich der Karte hinsichtlich des Pokémon Namen untersucht, indem der Bereich der Tesseract Engine als Input übergeben wird. Alle Wörter des extrahierten Texts werden isoliert und mittels Ähnlichkeitsberechnung mit allen Namen aus der Datenbank verglichen, welche der extrahierten ID zugeordnet sind. So wird bereits ein Großteil des zu betrachtenden Datensatzes ausgeschlossen. Für die Ähnlichkeitsberechnung wird die in Abschnitt 2.5.3 eingeführte Levenshtein-Distanz verwendet.

Hier ist die durchgeführte Partitionierung des Bildes entscheidet, da in der Bildbeschreibung oft von anderen Pokémon gesprochen wird, aus denen sich die aktuelle Karte entwickelt hat. Die Partitionierung lässt die Beschreibung weg und verhindert so bereits einige False Positives.

4.5 Teststrategie und Validierung

Nach der Einführung der zu betrachtenden Merkmalsextraktionsmethoden ist die Entwicklung einer Testkonzeption angebracht, um die Kombination der Vorverarbeitung mit den Extraktionsmethoden und Distanzmetriken auszuwerten. Die Pipeline wird dafür in die Setup- und Evaluationsphase aufgeteilt, die in Abbildung 4.7 visualisiert wird.

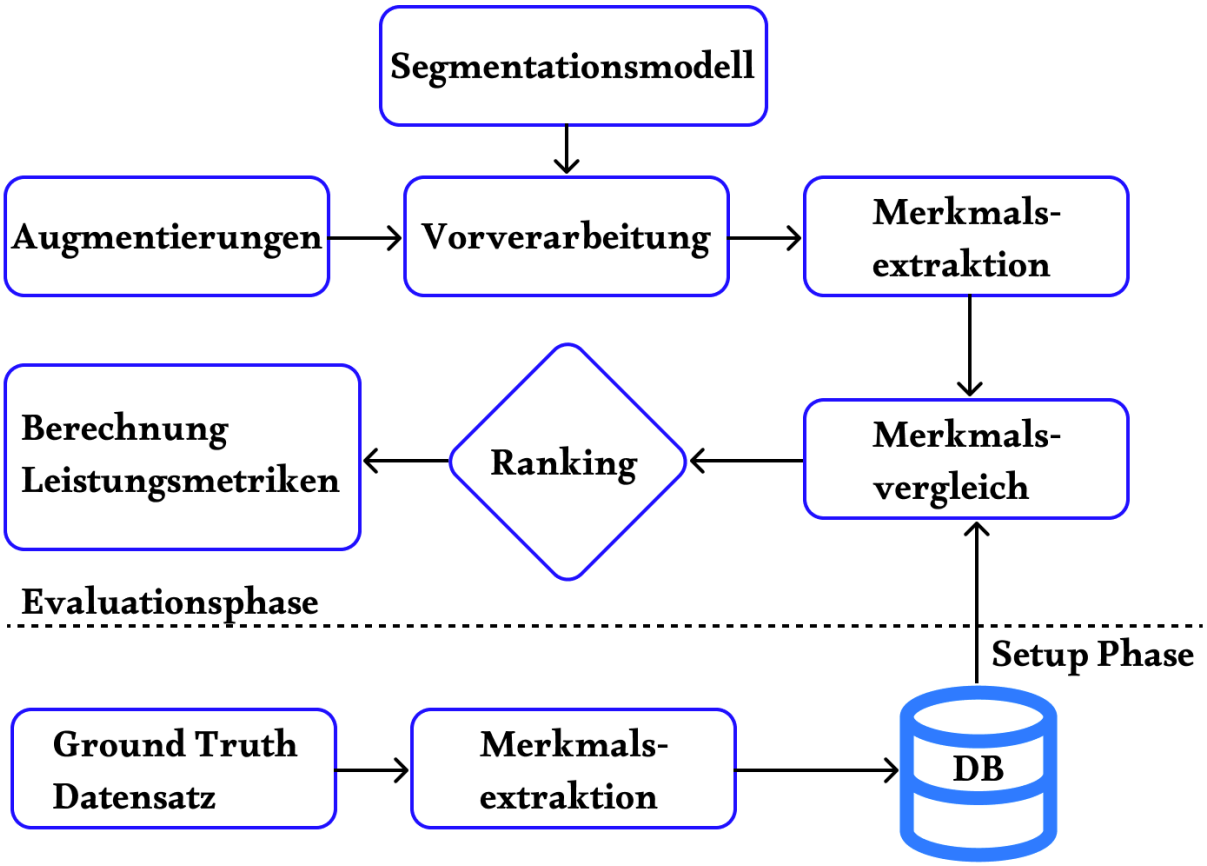


Abbildung 4.7: Test und Validierungspipeline

Da der Zweck dieser Arbeit sich mit der Forschungsfrage beschäftigt, welche Modifikationen an der Datenpipeline die besten Ergebnisse hinsichtlich der Kartenerkennung ermöglichen, muss ein Datensatz entwickelt werden, der die realen Gegebenheiten eines Kartenscan's simuliert. Zu diesem Zweck wird auf jede Karte eine simple und eine komplexe Augmentation durchgeführt, die eine Generalisierung des Datensatzes auf verschiedene Bildumgebungen ermöglicht. Dazu zählen verschiedene Hintergründe und Störeinflüsse, um das Datenset umfangreich genug zu gestalten. Als Grundlage für die Augmentationen werden die originalen Karten verwendet, die ohne Störungen und in hoher Kartenqualität vorliegen. Die genau Umsetzung und Erläuterung der Augmentationen erfolgt im Implementierungskapitel, Abschnitt 5.3.

Der Ablauf der beiden Testphasen wird im Folgenden näher beschrieben:

- Setup: Der Ground Truth Datensatz wird durchiteriert und für jede vorgestellte Merkmalsextraktionsmethode die benötigten Merkmalsvektoren aus den Bilddaten generiert. Da die originalen Bilder bereits

in der benötigten Vogelperspektive vorliegen, ist kein Vorverarbeitungsschritt nötig. Die Vektoren werden in einer Datenbank für den späteren Bildabruf gespeichert.

- Evaluationsphase: Die in der Augmentierung erstellten Datensätze für das simple und komplexe Testszenario werden im ersten Schritt Bild für Bild an die Vorverarbeitung übergeben. Dort werden die Karten aus dem Hintergrund extrahiert und in die gewollte Ausrichtung gebracht. Diese nun normalisierten Testbilder an die Merkmalsextraktion übergeben, wo je nach ausgewählter Methode die Merkmale des Bildes als Merkmalsvektor extrahiert werden. Im nächsten Schritt werden erfolgt der Merkmalsvergleich, bei dem die Testbilder jeweils mit allen Merkmalsvektoren der Datenbank mittels der in Abschnitt 3.4 vorgestellten Distanzmetriken verglichen werden.

Um die Pipeline hinsichtlich Ihrer Leistungsfähigkeit in Bezug auf unterschiedliche Datensatzgrößen zu untersuchen, werden die Szenarien in die drei Varianten unterteilt: 50, 500, 5000. Die für die Augmentationen verwendeten Bilder des originalen Datensatzes werden zufällig zu den Datensätzen hinzugefügt, um die Dominanz eines bestimmten Sets im Datensatz auszuschließen.

Um sinnvolle Leistungsmetriken für die Bewertung der gesamten Pipeline zu entwickeln, sind einige Faktoren zu beachten. Aus der Literaturanalyse zur Datenlage und Häufigkeitsverteilung in Abschnitt 3.5 ist erkennbar, dass einige wenige Pokémon öfter vorkommen als andere und dies bei der Klassenverteilung zu Problemen führen kann, wenn es um die Evaluierung der Klassifikationsergebnisse von Bildabrufsystemen geht. Da das Ziel des hier vorgestellten Konzeptes allerdings die Klassifikation aller Karten anhand Ihrer eindeutigen Merkmale ist und nicht der Kategorien, können Leistungsmetriken verwendet werden, die von ausgewogenen Klassen ausgehen. Für die Leistungsbewertung wird die durchschnittliche Klassifikationsgenauigkeit eingeführt, welche bereits für die Bewertung des Instanzsegmentierungsmodells herangezogen wurde. Des Weiteren ist eine Metrik erforderlich, die misst, ob nicht die ähnlichste, sondern gegebenenfalls die nächst ähnlichsten Karten die richtige enthalten. Um die gesamte Leistungsfähigkeit der Pipeline über alle Klassen hinweg zu messen, wird die durchschnittliche Klassifikationsgenauigkeit eingeführt:

$$\text{Durschnittliche Klassifikationsgenauigkeit} = \frac{\text{Anzahl korrekt klassifizierter Augmentationen}}{N} * 100,$$

wobei N die Gesamtanzahl an Klassen im Datensatz beschreibt. Sie ermöglicht eine ausgewogene Leistungsbewertung der Klassifikationsgenauigkeit über alle Klassen hinweg und so den Vergleich mit anderen Kombinationen innerhalb des Grundgerüsts.

Für die Messung, ob das richtige Bild in den nächst ähnlichsten Bildern enthalten ist, wird die Durchschnittliche Trefferquote in Fünf Versuchen (Bo5-Quote) eingeführt. Diese Metrik überprüft für jeden Durchlauf j von insgesamt k Durchläufen, ob das richtige Bild in den ersten fünf Ähnlichkeitswerten i enthalten ist. Die Summe dieser booleschen Variable, wird durch die Anzahl der Durchläufe geteilt, um die Bo5-Quote zu erfassen. Es gilt demnach:

$$X = \begin{cases} \sum_{j=1}^{k} \sum_{i=1}^{5} (P_i^{(j)} = \text{true}), & \text{if } P_i^{(j)} = \text{true}; \\ 0, & \text{if } P_i^{(j)} \neq \text{true} \end{cases} \tag{4.1}$$

, wobei $P_i^{(j)}$ die Bedingung P überprüft, ob das Bild in der i-ten Iteration der j-ten Wiederholung die richtige Klasse erkennt.

Die Bo5-Quote wird demnach berechnet aus:

$$\text{Bo5-Quote} = \frac{\text{X}}{N} * 100,$$

Nach der vollständigen Durchführung einer Iteration der Evaluationsphase werden die Leistungsmetriken bezüglich der vorgestellten durchschnittlichen Klassifikationsgenauigkeit und Bo5-Quote für das simple und das komplexe Testszenario berechnet. Die Bo5-Quote wird für die drei Datengrößen als Durchschnittswert ausgegeben, um einen einfachen Überblick zu gewährleisten. Die so erzeugten Daten werden hinsichtlich der Leistungsfähigkeit der Pipeline in Kapitel 6.2 ausgewertet und bewertet.

Kapitel 5

Implementation

In diesem Abschnitt wird die Implementation des vorgeschlagenen Konzepts aus Kapitel 4 ausführlich beschrieben.

5.1 Eingesetzte Software

Der für diese Arbeit verwendete Code wurde in der Python (Version: 3.12.3) Programmiersprache geschrieben. Dabei wurden verschiedenste Codebibliotheken und Werkzeuge verwendet.

OpenCV (v:4.10.0.84) Die Standardbibliothek [Its15] für die Entwicklung von Anwendungen im Bereich der Computergrafik. Unterstützt Python durch eine Interfaceimplementierung. Die bevorzugte Bibliothek vieler Forscher, da redundante Algorithmen und Methoden für die Bildverarbeitung nativ unterstützt werden. Für diese Arbeit war die Bibliothek in jedem Abschnitt der entwickelten Pipeline eingesetzt.

Pandas (v:2.2.3) Besonders im Bereich des Data Engineering hilfreiche Bibliothek [tea20], die für diese Arbeit für die Verarbeitung des Datensatzes Anwendung fand.

Numpy (v:1.26.4) Die Standardbibliothek [Har+20] für mathematische Operationen und die Erstellung von größeren Datenstrukturen. In dieser Arbeit wurde sie besonders in der Vorverarbeitung und der Merkmalsextraktion eingesetzt.

Pytesseract (v:0.3.13) Werkzeug aus dem Bereich der Optical Character Recognition Algorithmik [Kay07]. Mittels Tesseract wurde ID-basierte Text Embedding umgesetzt.

ImageHash (v:4.3.1) Bibliothek [Kra17] für die Berechnung von Signaturen über verschiedene Perceptual Hashing Algorithmen.

ultralytics (v:8.3.25) Python-Bibliothek [JQC23], welche vorgefertigte Pipelines für die Instanzsegmentierung, Objektdetektion und weitere Computer grafische Problemstellungen bereitstellt. Grundlage für die Implementation der Instanzsegmentierung dieser Arbeit.

Matplotlib / Pyplot (v:3.9.2) Bibliothek [Hun07] für die Erstellung von Visualisierungen mittels Python. Viele Abbildungen dieser Arbeit wurden über diese erstellt.

Colab Notebook Google Cloud Dienst für die Bereitstellung von GPU-Rechenkapazitäten. Für diese Arbeit wurden über den Dienst Rechenkapazitäten eingekauft, um das Modell für die Vorverarbeitungspipeline zu trainieren.

5.2 Codestruktur

Der für die Arbeit entwickelte Code ist in vier Teile partitioniert. Der Evaluations-Service ist die Steuerzentrale der Implementation. Die drei Phasen der Kartenvorverarbeitung, Merkmalsextraktion und des Distanzvergleich werden von ihm gesteuert.

- Der Kartenvorverarbeitungs-Service ist für die Verwaltung des vortrainierten Segmentierungsmodells, sowie die darauf folgende perspektivische Transformation zuständig.
- Der Merkmalsextraktions-Service greift auf die implementierten Algorithmen der jeweiligen Merkmale zu und wendet diese auf die vorverarbeiteten Bilder an.
- Im Distanzvergleichs-Service werden extrahierte Merkmale mit den jeweils ausgewählten Distanzmetriken miteinander verglichen.

Für die Evaluation einer Methode wird der Datensatz des gewählten Testszenarios an den Vorverarbeitungsservice übergeben und die Merkmalsextraktionsmethode bestimmt. Die erkannten und transformierten Karten werden an den Evaluations-Service zurückgegeben, welcher die Karten dem Merkmalsextraktions-Service übergibt. Der Extraktions-Service extrahiert die Merkmale der Karten und nutzt den Distanzvergleichs-Service, um die Karten mit der höchsten Ähnlichkeit in der Datenbank zu finden. Die Implementierung der drei Services wird in den weiteren Abschnitten dieses Kapitels beschrieben. Die Ergebnisse werden dem Evaluations-Service zurückgegeben, welcher die Evaluationskriterien aus Abschnitt 4.5 berechnet.

Das Training des Yolo Segmentierungsmodell fand über den Google Cloud Dienst Google Colab statt, welcher Cloud Kapazitäten für das Training des Modells zur Verfügung stellt. Die Implementierung des Trainings wird in Abschnitt 5.5 erläutert. Die durchgeführten Implementierungen der folgenden Kapitel werden, wo sinnvoll, mit Pseudocode annotiert.

5.3 Datensatzerstellung

Für die Implementierung dieser Arbeit wurden zwei thematisch unterschiedliche Datensätze erstellt. Für das Training des Segmentierungsmodells wurde ein Datensatz manuell gelabelt, welcher die äußeren Kartenränder der Pokémonkarten durch Bounding Boxes beschreibt. Diesen manuellen Kartenumrissen wurde der Metadateneintrag „TCG-Karte “zugewiesen, um die Kartenklasse zu beschreiben. Der entstandene Kartensatz

umfasst 163 manuell gelabelte Karten, die einzelne Pokémon-Karten, aber auch mehrere Karten und somit Klassen innerhalb der Bilder umfasst. Die Karten für diesen Datensatz wurden durch Onlinesuche, eigenes Scannen von Karten und mithilfe von Betreuern und Freunden des Autors dieser Arbeit zusammengetragen. Abbildung 5.1 zeigt ein Beispiel eines solchen manuellen Labelings. Die 163 Karten wurden, wie in der Analyse der Autoren von [GKK18] empfohlen, in 80 % Trainings und 20 % -Testdaten aufgeteilt.

Abbildung 5.1: Globales Labeling einer Karte

Da der Datensatz aus unterschiedlichen Quellen zusammengetragen wurde, weist er viele Störeinflüsse und Qualitätsunterschiede auf. Dies erhöht die Generalisierung des Modells und soll der Verbesserung der Erkennungsgenauigkeit dienen.

Der zweite thematische Schwerpunkt bei der Datensatzentwicklung befasste sich mit der Erstellung eines Argumentationsdatensatzes für die Evaluation der gesamten Pipeline. Im ersten Schritt musste ein Datensatz gefunden werden, der alle Pokémon-Karten repräsentiert. Unter den Gefundenen hat sich der Datensatz des Anbieters images.pokemontcg.io als am umfangreichsten herausgestellt. Er umfasst 13.140 Karten in der englischen Ausführung. Dieser wurde über ein eigens dafür entwickeltes Skript heruntergeladen und die Trainer und Energiekarten herausgefiltert.

Die Daten im Kartensatz wurden alle in hoher Qualität mittels eines Scanners aufgenommen, welche den Bildern eine Farbtiefe von mindestens 24 Bit verschafft. Allerdings sind die Kartenecken nicht gerundet, wie bei den originalen, was bei der Auswertung zu beachten ist. Für die Erstellung eines Argumentationsdatensatzes wird sich der Pipeline zur Datensatzaugmentation, welche in Abbildung 5.2 gezeigt wird, bedient. Zunächst werden zufällig Karten aus dem originalen Datensatz den Datengrößen: 50, 500, 5000 zugeteilt.

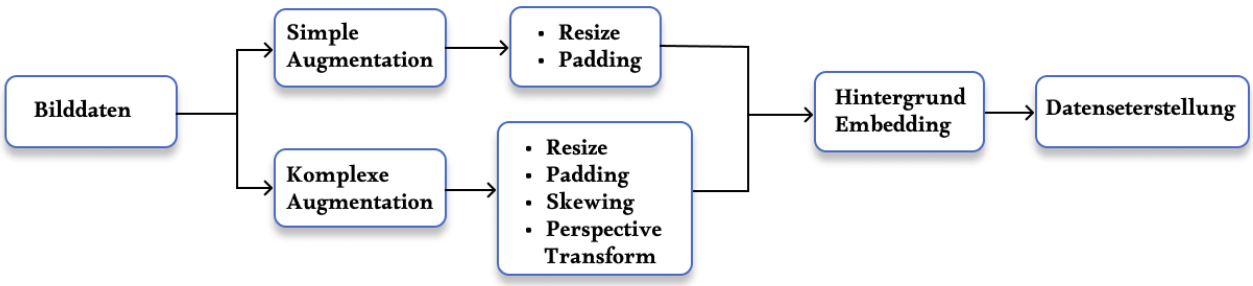
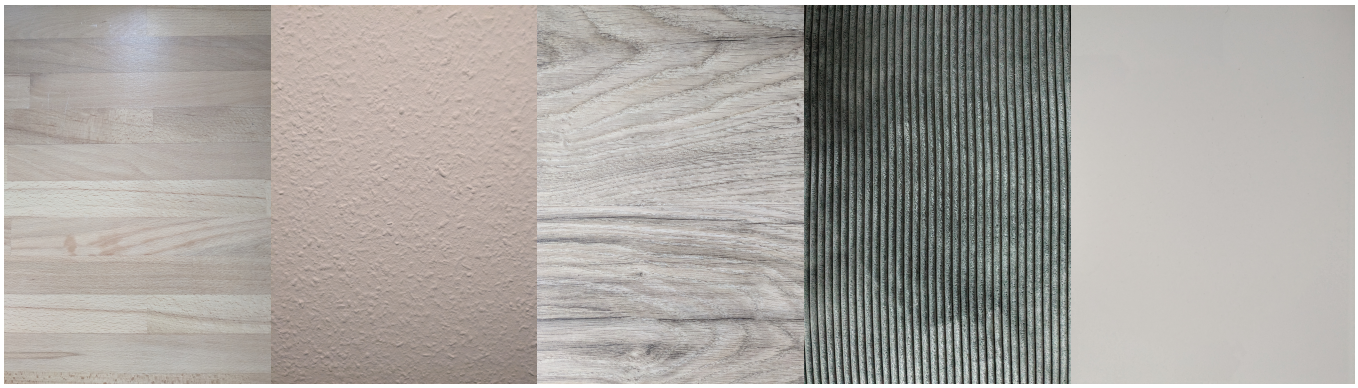


Abbildung 5.2: Pipeline zur Datensatzaugmentation

Es werden zwei Arten von Augmentation durchgeführt:

- Simple Augmentation: Die Pokémon Karte wird auf einen Hintergrund platziert, der typische Umgebungen darstellt.
- Komplexe Augmentation: Zunächst simple Augmentation. Perspektivische Transformation, wie in 2.3.4 vorgestellt, zur Seite, und oder hinten, um einen Kippeffekt zu simulieren. Jedes Pixel wird mit einer Wahrscheinlichkeit von 2% mittels Punktrauschen, wie in Abschnitt 2.3.1 vorgestellt, manipuliert, um tote Pixel, beziehungsweise schlechte Aufnahmequalität zu simulieren.

Die in Abbildung 5.3 gezeigten Hintergründe stellen typische Aufnahmeumgebungen von Sammelkartenbildern dar. Die Hintergründe wurden mit einer Auflösung von 3024 x 4032 Pixel und einer 24-Bit-Farbtiefe aufgenommen.



(a) Parkett (b) Raufasertapete (c) Laminat (d) Couch (e) Fliese

Abbildung 5.3: Hintergründe für die Augmentation

Das Augmentationsvorgehen wird daraufhin auf die Datensätze angewandt. So entstehen insgesamt drei simple sowie komplexe Datensätze. Der erzeugte Datensatz umfasst pro Szenario 5550 Bilder und somit insgesamt 11.100 Augmentierungen, die für die Evaluation verwendet werden. Abbildung 5.4 zeigt jeweils ein so erzeugtes Bild aus dem simplen und komplexen Szenario abgebildet.

(a) Simple Augmentation

(b) Komplexe Augmentation

Abbildung 5.4: Beispielaugmentationen

5.4 Training des Segmentierungsmodells

Für die Implementierung der Vorverarbeitung benötigt die Pipeline ein Segmentierungsmodell, welches mit hoher Präzision Karten von Hintergründen pixelgenau abgrenzen kann. In der Konzeption 4.3 wurde dafür das YOLOv11 Segmentierungsmodells ausgewählt. Dieser Abschnitt befasst sich mit der Umsetzung der Feinabstimmung des Modells.

Für das Training des Modells wurde Google Colab Research verwendet. Die Plattform erlaubt die Verwendung von T4 NVIDIA GPU Hardware. Das vortrainierte Modell yolo11n-seg.pt von Ultralytics wurde für die Feinabstimmung verwendet. Das Training erfolgte auf den in 5.3 erstellten Datensatz, welcher auf 130 Trainings und 33 Validierungsbildern basiert. Die Implementierung der Feinabstimmung ist durch das Ultralytics Traningsframework vorbestimmt und muss lediglich bezüglich einiger Konfigurationsentscheidungen angepasst werden.

Trainingskonfiguration

- Anzahl an Trainingsepochen: 20
- Bildgröße: 640 × 640

Die Evaluation des Modells erfolgt in Abschnitt 6.1.

5.5 Vorverarbeitung

Die Vorverarbeitung der TCG Karten erfolgt anhand der Konzeption der Vorverarbeitungsstrategie aus Abschnitt 4.3.

Die Vorverarbeitungsimplementation teilt sich in drei Hauptschritte ein.

- Schritt 1: Klassenerkennung durch das fein abgestimmte Segmentierungsmodell.
- Schritt 2: Approximation und Zuschneiden der Karte anhand der Segmentierungsmaske.
- Schritt 3: Perspektivtransformation zur Standardisierung der Kartenansicht.

Der erste Schritt der Vorverarbeitung befasst sich gänzlich mit der Erkennung der Kartenklasse: „Sammelkarte "innerhalb eines Eingabebildes. Das Eingabebild wird dem vortrainierten Segementierungsmodell aus 5.4 als Eingabe übergeben. Dafür wird ein Eingabebild an die Prediction Factory des Open source Paketes Ultralytics übergeben, welches die Gewichte des feinabgestimmten Modells als Eingabeparameter für das Yolo Segmentationsmodells nutzt. Das Vorhersagemodell erstellt zwei Ausgaben. Die Bounding-Box-Koordinaten beschreiben mittels eines Rechtecks, die vorhergesagten Kartenbereiche, während die Segmentierungsmaske eine pixelgenaue Maske über das Eingabebild liegt, um die Klasse zu beschreiben. Der in Abschnitt 4.3 erstellten Konzeption folgend ist die Maske wesentlich genauer für die Approximation der Kartenränder, da die Segmentierungsmaske pixelgenau nur die Kartenklasse markiert.

Im zweiten Schritt muss nun die Kontur eine rechteckige Form approximiert werden. Mittels des OpenCV Packets wird zunächst der Perimeter einer Kontur berechnet. Der Parameter ϵ beschreibt dabei die Approximationsstoleranz mittels eines Prozentanteils des Umfangs der Kontur. Ein Größeres ϵ sorgt für das Weglassen von mehr Details und kann so für Merkmalsverlust führen. Je kleiner das ϵ gewählt wird, desto mehr Details bleiben erhalten. Da es sich bei TCG Karten um Rechtecke handelt, können alle Konturen mit mehr oder weniger Eckpunkten fallengelassen werden. Da die Kanten bei der Generierung durch das Yolo Modell meist bereits annähernd gerade sind, ist keine große Vereinfachung notwendig. Ein zu großes ϵ könnte weitergehend aus einem Rechteck ein Trapez generieren. Daher wird der Parameter hier auf einen kleinen Wert von 2% des Perimeters gesetzt, um dies zu vermeiden.

Die Durchführung der Approximation erfolgt durch die openCV Funktion approxPolyDP, welche mittels der gegebenen Konturen und dem erläuterten Epsilon Parameter Rechtecke erzeugt.

Die erhaltene Approximation enthält die vier Eckpunkte der Karte. Allerdings sind diese möglicherweise in der falschen Reihenfolge angeordent. Die in Abschnitt 4.3 vorgestellte Sortierung der Eckpunkte erfolgt mittels der folgenden Logik:

- Links-Oben: Punkt mit dem kleinsten x-y Wert.
- Links-Unten: Punkt mit dem größten x und kleinsten y Wert.

- Rechts-Oben: Punkt mit dem größten x und y Wert.
- Rechts-Unten: Punkt mit dem kleinsten x und größten y wert.

Diese Sortierung ist wichtig, da die nun folgende Perspektivtransformation die Eckpunkte in der vorgestellten Reihenfolge benötigt.

Für die Durchführung der perspektivischen Transformation wird die Durchführung des Homografieverfahrens, wie in Abschnitt 2.3.4 vorgestellt, verwendet.

Für die Perspektivtransformation wird das Homografieverfahren verwendet, welches bereits in Abschnitt 2.3.4 vorgestellt wurde, um die Verzerrung zu korrigieren und das Trapez in ein Rechteck zu überführen. Dafür wird die Homografiematrix H berechnet. Die Bildkoordinaten (x, y) eines Eingabebildes sollen in die Zielkoordinaten (x', y') transformiert werden. Dafür wird jedes Pixel der approximierten Kontur auf das entsprechende Pixel im Zielrechteck abgebildet, welches die zu erreichend Vogelperspektive darstellt.

Die resultierende Matrix H wird dann auf jedes Pixel des Trapezes angewendet, um die endgültige, entzerrte Perspektive zu erzeugen. Für die Umsetzung wird die OpenCV Funktion getPerspectiveTransform verwendet. Diese nimmt als Eingabeparameter die Zielkoordinaten der gewünschten Vogelperspektive und führt die Ebenenprojektion durch.

5.6 Merkmalsextraktion

Der Vorverarbeitung folgend werden in diesem Kapteil die Implementationen der Merkmalsextraktion vorgestellt. Diese sind in die Unterabschnitte für Histogramme 5.6.2, Hashing Methoden und 5.6.3 Texterkennung 5.6.2 aufgeteilt.

5.6.1 Implementation der Histogrammbasierten Methoden

Histogram of oriented Gradients

Die Implementierung der Histogram of Oriented Gradients Methodik wird im folgenden anhand der Ausführungen der Autoren von [Hal13]erläutert. Der erste Schritt der Implementierung ist die Berechnung der Gradientenwerte. Dafür muss in vertikaler und horizontaler Richtung die Ableitungsmaske mittels eines Filterkerns angewendet werden. Die Autoren verwenden den in Abschnitt 2.3.2 bereits vorgestellten zentralen Differenzoperator für diese Operation. Dieser wird für die horizontale DX, sowie vertikale Ableitung DY definiert: $DX = [-1, 0, 1]$, $DY = [-1, 0, 1]$.

Auf ein Eingabebild I wird die Faltungsoperation in x- und y-Richtung berechnet: $I_X = D * X$, $I_Y = D * X$. Der Gradientenbetrag kann nun, wie in 2.6 vorgestellt für I_x und I_Y berechnet werden als: $G = \sqrt{I_x^2 + I_y^2}$

Schließlich ergibt sich die Gradientenorientierung, wie in Gleichung 2.4, vorgestellt für die Bildgradienten: $\theta = \arctan\left(\frac{I_Y}{I_X}\right)$. Die Grundidee für die Verwendung von HOG ist Beleuchtungsinvarianz zu erreichen. Dafür verwenden die Autoren ein $8*8$ Nachbarschaftsfenster (Zelle). Diese iteriert über das Bild und füllt die Bins, welche die Orientierungswinkel θ beschreiben, mit den berechneten Gradienten G jeder Bildkoordinate. Nach den Autoren liefern 9 Orientierungen dabei die besten Ergebnisse.

Um Beleuchtungsänderungen auszugleichen, verwenden die Autoren Blöcke, welche mehrere Zellen kombinieren und die Zellhistogramme normalisieren. Im Paper zwar nicht explizit erwähnt, führt dies auch zu einer Invarianz gegenüber Rotationen.

Die Bibliothek OpenCV hat besitzt bereits eine Implementierung des HOG Deskriptors, wodurch eine eigene Implementierung entfällt. Die grundlegende Funktion des Deskriptors, wie oben erläutert, wird anhand des Pseudocodes in Algorithmus 11 nachgestellt.

Algorithm 1 Berechnung des HOG

```
function ComputeHOG(I, bins = 9)
    G_X, G_Y ← ComputeGradients(I)
    G, θ ← ComputeMagnitudeAndOrientation(G_X, G_Y)
    for each cell c in I do
        H(c) ← ComputeHistogram(G, θ, bins = 9)
    end for
    for each block b in I do
        H(b) ← NormalizeBlock(H(c))
    end for
    return H
end function
```

Als Ergebnis wird der HOG Feature Vektor H zurückgegeben.

Implementation des Local Binary Pattern

Die Autoren von [Xia+21] geben keine direkte Codeimplementation, jedoch eine mathematische Beschreibung des Local Binary Pattern Verfahrens, welches für eine eigene Implementation verwendet wird. Das Grundprinzip des Verfahrens wurde bereits in 2.5.2 vorgestellt. Die Grauwerte der neun Pixel im Nachbarschaftsfenster werden mit dem Grauwert des Pixels in der Mitte verglichen. Falls deren Werte größer als der Mittelpunkt sind, werden Sie mit einer 1 im Fenster markiert. Die Binärwerte der 8 Pixel werden für den gerade betrachteten Block als dessen Sequenz zusammengefasst und in einer Binärmatrix M für alle Iteration über das Bild gespeichert. Wie in 2.5.2 visualisiert werden aus den Sequenzen Dezimalwerte berechnet und diese im Ausgaberaum mittels eines Histogramms gespeichert, welches die Kanäle 0-256 aufweist. Der Vorgang der Implementierung wird anhand des Pseudocodes in Algorithmus 2 dargestellt.

Algorithm 2 Berechnung des Local Binary Pattern

```
1: function ComputeLBP(image)
2:     gray_img ← ConvertToGrayscale(image)
3:     for (x, y) ∈ pixels of gray_img do
4:         neighbors ← GetNeighborhood(gray_img, (x, y))
5:         LBP(x, y) ← CalculateBinaryMatrix(neighbors)
6:     end for
7:     return LBP
8: end function
```

Implementation des HSV Hue Histogramm

Die Autoren von [DKN08] verwenden für die Merkmalsextraktion den RGB Farbraum. Wie bereits in 4.4 argumentiert wurde, eignet sich dieser nicht gänzlich für die unterschiedlichen Lichtverhältnisse, die bei den hier betrachteten Eingabebildern Szenariobildern vorliegen. Daher wird das gleiche Prinzip der Autoren statt auf den RGB Farbraum auf den HSV Farbraum angewandt, welcher die Helligkeit im separaten Value Kanal speichert und so die Sättigung und den Farbton des Eingabebildes nicht beeinflusst. Für die eigene Implementation wird mittels OpenCV der Hue Channel in 360 Bins unterteilt. Dementsprechend erhält jeder Winkel der Grundfarbe im Wertebereich von 0° bis 360° einen eigenen Kanal. Der Algorithmus 5 wird als Pseudocode repräsentiert.

Algorithm 3 Helligkeitsbasiertes Histogramm

```
1: function ComputeHueHistogramm(I)
2:     Y_Channel ← ExtractYChannel(I)
3:     hist_Y ← CalculateHist(Y_Channel)
4:     return NormalizeHistogramm(hist_Y)
5: end function
```

5.6.2 Implementierung der Hash basierten Methoden

In diesem Unterabschnitt werden die Implementierungen der Hash basierten Merkmalsxtraktionsmethoden hinsichtlich Ihrer Unterschiede und Ähnlichkeiten vorgestellt, um dem Leser einen Überblick über die Verfahren zu geben. Dabei wird sich, wo vorhanden, anhand der Beschreibung der Implementierung der jeweiligen Veröffentlichungen orientiert. Falls diese nicht vorhanden sind, oder schlecht beschrieben, erfolgt eine eigene Implementation. Methoden, die einen binären Ausgaberaum verwenden, skalieren diesen auf 128 Bit, um, wie in Abschnitt 4.4.2 erläutert, Kollisionen zu verhindern.

Implementation des Luminanzbasierten DCT-Hashing

Das originale Perceptual Hashing Paper von [Zau10] vergleicht DCT basierte Hash Werte mit Marr-Hildreth Operator basierten. Die Besonderheit des Ansatzes ist die Umwandlung der Bildpixel in Ihre Grauwerte auf

Grundlage von dessen Leuchtdichte. Der Autor argumentiert, dass die essenziellen Informationen nur anhand der Leuchtdichte beschrieben werden können[Zau10, S.30]. Wie anhand der Ergebnisse anderer vorgestellter Arbeiten zu erschließen, gibt es allerdings viele weitere Merkmale, die ähnlichen Ergebnisse mittels anderer Farbräume erzielen. Die Ausführungen des Autors wurden in der Python Bibliothek ImageHash [Kra17] mittels Python implementiert und bedarf daher keiner eigenen Implementation. Das Grundprinzip des Verfahrens ist, das Bild zunächst in Grauwerte umzuwandeln, um daraufhin DCT auf die Zeilen und Spalten der Pixelwerte anzuwenden. Laut den Autoren befinden sich die relevantesten niedrigfrequenten Merkmale, die bereits das gesamte Bild beschreiben, in der oberen linken Hälfte der DCT Matrix. Der Hintergrund dieser Annahme ist die Eigenschaft der niedrigrequenten Merkmale, welche im Abschnitt zu DCT 2.3.3 bereits vorgestellt wurden. Das verwendete Nachbarschaftsfenster für diese Extraktion orientiert sich an der Größe des Ausgaberaums N. Der Autor verwendet ein 8x8 Fenster, welches 64 DCT Koeffizienten erzeugt. Diese werden in ein eindimensionales Array überführt, aus dessen der Median m berechnet wird. Das Array der Koeffizienten wird nun durchiteriert und den DCT Koeffizienten an der Stelle C_i entweder 0 oder 1 zugewiesen. Der erzeugte Hash ist im Vergleich mit den anderen in diesem Kapitel implementierten Verfahren der einzige, der einen binären Ausgaberaum erzeugt.

$$h_i = \begin{cases} 0, & \text{if } C_i < m, \\ 1, & \text{if } C_i \geq m, \end{cases} \quad \text{for } i = 1, 2, \ldots, 64$$

Die Schritte werden im Pseudocode des Algorithmus 4 abgebildet.

Algorithm 4 Helligkeitsbasiertes-Hashing

```
1: function pHash(image, hash_size = 128, N = 8)
2:     gray_img ← ConvertToGrayscale(image)
3:     dct_matrix ← DCT(gray_img)
4:     dctlowfreq ← SubMatrix(N, N, dct_matrix)
5:     m ← Median(dctlowfreq)
6:     H ← calculateBinaryHash(dctlowfreq, m)
7:     return H
8: end function
```

Implementation des Ringpartitions Hashing

Das Paper [Tan+16a] beinhaltet keine direkte Codeimplementation des Algorithmus. Für diese Arbeit wird auf Grundlage der Erläuterungen der Autoren ein eigener umgesetzt. Das Grundprinzip des Algorithmus ist durch die Nutzung von Ringpartitionen Rotationsrobustheit einzuführen. Außerdem wird mittels des Gaußschen Filter eine Rauschunterdrückung durchgeführt. Die Autoren verwendeten für Ihre Experimente eine Skalierung von 512x512, dies wird für die eigene Implementierung aus Speicher und Rechenkapazitätsgründen auf 256x256 reduziert. Im ersten Schritt wird dafür das Eingabebild auf 256x256 Auflösung skaliert und daraufhin der Gaußsche Filter, welcher in Abschnitt 2.3.1 vorgestellt wurde, angewandt. Das skalierte Bild wird in den, in Abschnitt 2.2.1, vorgestellten Lab Farbraum L^* konvertiert.

Nach der Vorverarbeitung berechnen die Autoren gleich große Ringpartitionen über das vorverarbeitete Bild. Dafür werden vier konzentrische Ringe mit gleicher Fläche festgelegt, dessen Pixel bei einer Rotation des Bildes unverändert bleiben. Dies ist die Grundidee der Rotationsrobustheit der Autoren. In Ihren Experimenten untersuchten die Autoren, welchen Einfluss die Veränderung der Anzahl von Ringpartitionen auf die Klassifikationsgenauigkeit hat. Sie kamen zu zu dem Schluss, dass 40 Ringe die besten Klassifikationsergebnisse liefern. Daher wird diese Anzahl auch in dieser Implementation verwendet. Alle Experimente wurden mittels eines Nachbarschaftsfenster in der Größe 3x3 durchgeführt. Dies wird von dieser Implementierung in Algorithmus 8 übernommen.

Algorithm 5 Ring-Partitionierung

function PARTITIONINTORINGS($L^*, n = 40, N = 256$)
    $center \leftarrow (N/2, N/2)$
    $radii \leftarrow ComputeRingRadii(N, n)$
    for $k = 1$ to n **do**
        $R_k \leftarrow ExtractRingPixels(L^*, center, radii[k])$
    end for
    return R
end function

Da die Ringpartitionen selbst noch nicht als Feature ausreichen, werden für jeden Ring die folgenden statistischen Merkmale berechnet: Mittelwert, Varianz, Verzerrung, Wölbung. Aus diesen wird eine Merkmalsmatrix extrahiert, welche die statistischen Merkmale jedes Rings ausdrückt. Dieser Schritt ist in Algorithmus 11 abgebildet.

Algorithm 6 Merkmalsextraktion aus den Ringen

function EXTRACTFEATURES($R, n = 40$)
    for $k = 1$ to n **do**
        $\mu_k \leftarrow$ ComputeMean(R_k)
        $\delta_k \leftarrow$ ComputeVariance(R_k)
        $s_k \leftarrow$ ComputeSkewness(R_k)
        $w_k \leftarrow$ ComputeKurtosis(R_k)
        $v_k \leftarrow (\mu_k, \delta_k, s_k, w_k)$
    end for
    $V \leftarrow [v_1, v_2, ..., v_n]$
    return V
end function

Der letzte Schritt umfasst die Hash-Generierung aus den Vektoren der Merkmalsmatrix. Dafür wird zunächst ein Referenzvektor aus dem Mittelwert der Spalten der Merkmalsmatrix berechnet. Nun wird der euklidische Abstand zwischen jedem Ring Vektor und dem Referenzvektor berechnet. Da der berechnete Abstand $d(i)$ eine Gleitkommazahl ist, schlagen die Autoren die Quantisierung auf eine ganze Zahl vor. Zurückgegeben wird ein finaler Hash H.

Algorithm 7 Hash-Generierung aus der Merkmalsmatrix

function GENERATEHASH($V, n = 40$)
  $Q \leftarrow NormalizeFeatureMatrix(V)$
  $c_{ref} \leftarrow ComputeReferenceVector(Q)$
  for $i = 1$ to n **do**
    $d(i) \leftarrow ComputeEuclideanDistance(Q[i], c_{ref})$
    $z(i) \leftarrow QuantizeDistance(d(i))$
  end for
  $H \leftarrow [z_i, z_1, ..., z_n]$
  return H
end function

Da die Forscher in Ihrer Untersuchung auch die mögliche Manipulation des Hashwertes betrachten haben, wurde eine Permutation des Hashwert eingeführt, um Manipulationen zu verhindern. Da dies für unsere Implementation allerdings nicht relevant ist, entfällt diese in der eigenen Implementation.

Farbvektorbasiertes Hashing

Die Autoren von [Tan+18] geben in Ihrem Paper keine direkte Implementation des Verfahrens, jedoch eine gute Konzeption des Prinzips wieder. Aus dieser wird folgend eine eigene Implementation abgeleitet. Das Grundprinzip des vorgestellten Verfahrens ist die Extraktion des Farbwinkels als Merkmal eines Bildes. Diese Methode wurde von den Autoren vorgeschlagen, da Sie argumentieren, dass Forschung im Bildabruf oft die Extraktion der Helligkeit (Luminzanz) aus dem HSV-Farbraum verwendet und diese keine gute Repräsentation der Farbverteilung eines Bildes ermöglicht. Da die Extraktion von Farbverteilung, beispielsweise mittels des RGB-Farbraums, bei Augmentationen des Originals zu Merkmalsverlust führen kann, schlagen die Forscher den Farbwinkel als Extraktor vor. Dieser ist invariant gegenüber Intensitätsänderungen, durch seine Repräsentation von Farbverteilung, sowie Luminanz. Die Autoren verwenden zunächst einen Gaußschen Filter, dessen Prinzip in Abschnitt 2.3.1 vorgestellt wurde, um das Bildrauschen zu reduzieren. Dafür wird das Bild zunächst auf 256 x 256 skaliert und mittels eines 5x5 Nachbarschaftsfensters der Filter über das Bild iteriert. Für jedes Pixel des Bildes wird im Eingabebild ein RGB-Vektor $V_r = [R, G, B]$ extrahiert, wobei i die Anzahl der Pixel im Bild repräsentiert. Da für die Berechnung des Farbwinkels ein zweiterer Referenzvektor benötigt wird, werden die Durchschnittswerte der Farbkomponenten im Vektor $V_2 = [R_r, G_r, B_r]$ gespeichert. Für jedes Pixel des Bildes wird nun der Sinus-Wert des Farbwinkels berechnet, indem die Differenz zwischen den RGB-Durchschnittswerten V_2 und den Farben des jeweils betrachteten Pixels von V_r berechnet werden: $\sin\theta = \sqrt{1 - \frac{(V_r^T V_2)^2}{V_r^T V_r V_2^T V_2}}$. Nach Durchführung der Berechnung für jedes Pixel wird die entstandene Merkmalsmatrix A verwendet, um daraus ein Histogramm zu erstellen. Um zusätzliche Rotationsrobustheit einzuführen, schlagen die Autoren eine Kreisregion, um den Mittelpunkt des Bildes vor, aus dem die Häufigkeitsverteilung berechnet wird. Der Ausgaberaum der Sinus-werte liegt zwischen 0 und 1. Um aus diesem Raum ein Histogramm zu bilden, schlagen die Autoren eine Abtastung des Wertebereichs mittels einer 0.005er-Schrittgröße vor, was insgesamt 200 Bins für die Histogrammerstellung aus dem Kreisbereich

ermöglicht. Dies wird in Algorithmus 9 visualisiert. Die berechneten Sinus-Farbwinkel werden diesen Bins zugeordnet und das entstandene Histogramm normalisiert, damit die Summe über alle Bins 1 ergibt.

Algorithm 8 Histogramm Extraktion

```
1: function ExtractHistogram(A)
2:     Initialize histogram as empty
3:     for all (i, j) ∈ A do
4:         if pixel (i, j) is inside the circle then
5:             Update histogram with A[i, j]
6:         end if
7:     end for
8:     return Normalize(histogram)
9: end function
```

Im letzten Schritt verwenden die Autoren drei verschiedene Methoden für die Hash-Generierung. Um die Implementierung mit den anderen implementierten Methoden vergleichbar zu halten, beschränkt sich diese Arbeit auf das DCT-Prinzip, welches in Abschnitt 2.3.3 vorgestellt wurde. Das DCT-Verfahren wandelt das Histogramm der Farbwinkelwerte in den Frequenzbereich um und extrahiert die signifikantesten niedrigfrequenten DCT-Koeffizienten. Diese Koeffizienten werden dann quantisiert, um den Hash zu berechnen.

Strukturmerkmalsbasiertes Hashing

Das Paper der Autoren von [Qin+16a] beschäftigt sich mit der Entwicklung eines robusten Hash Algorithmus, der resistent gegenüber Rauschen und kleinen geometrischen Veränderungen ist. Der Hauptunterschied zu den andere hier implementierten Methoden ist die Verwendung von Kantenerkennung mittels des Canny-Edge-Algorithmus, welcher bereits in Abschnitt 2.4.2 vorgestellt wurde. Das vorgeschlagene Verfahren extrahiert kantenreiche Regionen des Bildes, aus denen mittels DCT mehrere dominante Bildblöcke extrahiert werden. Zusätzlich verwenden die Autoren eine Hauptkomponentenanalyse, um die Dimensionalität der Merkmale weiter zu reduzieren. Da dieser allerdings die Vergleichbarkeit mit den anderen implementierten Methoden verhindert, wird stattdessen die DCT Methodik, wie in Abschnitt 2.3.3 vorgestellt, verwendet. Für die Rauschunterdrückung wird aus dem Eingabebild zunächst der Luminanzkanal Y Kanal extrahiert. Es folgt eine Skalierung auf $256x256$ und die Anwendung eines $5x5$ Gaußschen Filters. Das gefilterte Bild wird in Algorithmus 9 hinsichtlich seiner Kanten mittels des Canny Operators mit einem unteren Schwellenwert von 50 und oberen von 150 untersucht. Das Verfahren teilt sich demnach in zwei Phasen auf. Zunächst werden mittels des Canny Edge Operators die Kanten des Bildes extrahiert. In der zweiten Phase werden mittels DCT die wichtigsten niedrigfrequenten Koeffizienten der Kantenregionen extrahiert. Das vorgeschlagene Verfahren verwendet dabei für die Kantendedektion die Luminanzinformationen des Y Kannals im HSV Farbraum. Die durch den Canny Algorithmus extrahierten Kantenregionen werden in Blöcke unterteilt, wobei die Autoren eine Blockgröße mit dem Parameter $k = 8$ festgelegt haben. Dies führt zu einer Unterteilung des Bilders in $8 * 8 = 32$ Blöcke. Es wird argumentiert, dass die Aufteilung in Blockregionen die Erfassung der wichtigsten Merkmale des Bildes erleichtert. Die Blöcke liegen als binäre Matrix vor, die beschreiben, ob ein Pixel in der

Kantenregion liegt oder nicht. Da die Dimensionalität durch die Anzahl der Kantenregionen noch zu hoch ist, schlagen die Autoren das selektive Sampling vor. Dabei werden die Einsen jedes Blocks in der binären Matrix gezählt und so die N Blöcke mit der höchsten Kantendichte ausgewählt. Als Anzahl der ausgewählten Blöcke schlagen die Autoren $N = 50$ vor, was für den späteren Vergleich mit der von Ihnen erreichten Baseline beibehalten wird. Die Berechnung der DCT Koeffizienten wird analog der Implementation 5.6.2 durchgeführt.

Dabei wird davon ausgegangen, dass die dominierenden niedrigfrequenten Koeffizienten DCT Koeffizienten die Bildregionen ausreichend beschreiben. Die Aufteilung Extraktion der Kantenregionen und die DCT Extraktion wird in Algorithmus 9 abgebildet.

Algorithm 9 Kantendedektion

1: **function** DETECTSALIENTEDGES($image, low\_thresh, high\_thresh$)
2: $edge\_map \leftarrow$ CANNY($image, low\_thresh = 150, high\_thresh = 150$)
3: $blocks \leftarrow$ DIVIDEINTOBLOCKS($edge\_map, N$)
4: $densities \leftarrow$ COMPUTEEDGEDENSITIES($blocks$)
5: **return** $CalculateDCT(densities)$
6: **end function**

Als letzten Schritt wird ein finaler Hashwert erzeugt, der das auf den DCT Koeffizienten der Kantenregionen basiert. Dafür wird zunächst die DCT Matrix M erstellt

5.6.3 Implementierung der Texterkennung

Die Implementierung der Konzeption zur Texterkennung orientiert sich am Paper: „Content based Image Retrieval using Tesseract OCR Engine and Levensthein Algorithm“[PT12], wurde jedoch für diese Arbeit auf Grundlage der eigenen Konzeption aus Abschnitt 4.4.3 angepasst. Das Eingabebild wird zunächst in zwei Partitionen unterteilt. In der oberen Partition befindet sich der Name und in der unteren die Seriennummer.

Dafür wird die Python Bibliothek Tesseract-OCR verwendet. Das Eingabebild wird zunächst in den in konvertiert. Das Bild wird nach der erfolgreichen Vorverarbeitung, durch die Das Verfahren zur Identifikation von Pokémon-Sammelkarten basiert auf Optical Character Recognition (OCR) und einer anschließenden Ähnlichkeitsanalyse. Zunächst wird das Kartenbild vorverarbeitet und in einen oberen und mehrere skalierten untere Segmente partitioniert. Der obere Bereich enthält den Namen des Pokémon, während die Set-Nummer im unteren Bereich lokalisiert wird. Die Tesseract-OCR-Engine wird zur Texterkennung verwendet.

Da die Seriennummer meist eine sehr kleine Schriftgröße folgt, wird die untere Partition mit einem Faktor von + 0.1 pro Iteration vergrößert. Falls nach fünf Iterationen keine Seriennummer erkannt wurde, wird die Karte als nicht erkannt markiert.

Die Extraktion der Seriennummer erfolgt anhand einer Regular Expression, wie in 4.4.3 vorgestellt. Die extrahierte Set-Nummer wird mit einer Datenbank abgeglichen, um eine Liste potenzieller Kartenkandidaten zu

erhalten. Anschließend wird der extrahierte Name mit den Namen der Kandidaten anhand der Levenshtein-Distanz verglichen, um das bestmögliche Matching zu ermitteln. Der detaillierte Ablauf dieses Verfahrens ist in Algorithmus 10 dargestellt.

Algorithm 10 OCR Tesseract basierte Merkmalsextraktion

```
1:  function FetchNameSetIDCombination(I)
2:      I_grau ← ConvertToGrayScale(I)
3:      U, L_scaled ← PartitionImage(I_grau)
4:      Name ← OCRImage(L_scaled)
5:      L_BestMatch ← ∅
6:      for L ∈ L_scaled do
7:          ExtractedText ← OCRImage(L)
8:          SetID ← ExtractSetID(ExtractedText)
9:          Similarity ← MatchCards(SetID)
10:         if Similarity < L_Matches then
11:             L_Bestmatch ← SetID
12:         end if
13:     end for
14:     return FetchCombinedVector(L_BestMatch, Name)
15: end function
```

5.6.4 Wahl und Implementierung der Distanzmetriken

Die in Abschnitt 5.6 beschriebenen Implementierungen liefern unterschiedliche Ausgabeformate. In Abschnitt 2.5.3 wurden Distanzmetriken beschrieben, welche sich für unterschiedliche Datentypen eignen.

Die Implementierungen geben jeweils drei Ausgabeformate vor, anhand derer die richtige Distanzmetrik gewählt werden muss. Einerseits werden normalisierte Histogramme ausgegeben, welche eine diskrete Verteilung der Eigenschaften der jeweiligen Methoden beschreiben. Diese normalisierten Vektoren enthalten jeweils so viele Werte, wie Bins für die Methode vorgegeben waren. Bins beschreiben hierbei die numerische Repräsentation eines Merkmals. Dieses Ausgabeformat wird einerseits von den Histogrammbasierten Methoden aber auch von den Hash basierten Methoden erzeugt, da das Ausgabeformat in beiden Fällen ein normalisierter eindimensionaler Vektor der Größe N darstellt, wobei N die Anzahl der Bins der Methode beschreibt, die für die Ausgabe normalisiert wurden. Für den Distanzvergleich sollen Methoden gewählt werden, die unterschiedliche Eigenschaften gegenüber den Merkmalsverteilungen aufweisen. Zunächst ist die Euklidische Distanz zu betrachten, welche die Differenzberechnung zweier Vektoren quadriert und so große Unterschiede zwischen Vektoren hervorhebt. Dies macht den Match von sehr unähnlichen Bildern unwahrscheinlicher, führt allerdings auch dazu, dass die Metrik für Ausreißer in den Merkmalen sensitiver ist.

Die Manhattan-Distanz ist wiederum robust gegenüber Ausreißern, da die Differenzen linear summiert werden und keine Quadrierung stattfindet. Jedoch fehlt somit auch die Empfindlichkeit gegenüber

Ausreißern. Bei beiden Metriken fallen besonders die Bins mit hohen Werten ins Gewicht. Da allerdings auch Bins mit kleinen Werten wichtige Merkmalsträger sind, ist eine Metrik, die diese stärker wertet, wichtig. Als weitere Ergänzung wird die Chi-Quadrat-Distanz eingeführt. Diese betont durch die Division durch $x_i + y_i$ kleinere Bins stärker als größere bei der Summierung. Seltene Merkmale werden so hervorgehoben.

Da die drei Metriken jeweils Schwächen und Stärken aufweisen, werden alle drei für die Auswertung der eindimensionalen Vektoren als Distanzmetrik verwendet.
Die Auswahl von Distanzmetriken für die anderen beiden Ausgabeformate ist wesentlich trivialer. Die String basierte Texterkennung wird mittels der Levenshtein-Distanz auf Ihre Ähnlichkeit zu anderen Strings hin untersucht. Binärwerte können mittels der Hamming-Distanz auf die nötige Anzahl an Stellen untersucht werden, an denen sich zwei Zeichenketten unterscheiden. Die Anzahl der nötigen Änderungen kann als Ähnlichkeitsmaß verwendet werden.
Die Ausgabeformate der implementierten Methoden sind in Tabelle 5.1 den jeweiligen Distanzmetriken zugeordnet. Ausschlaggebend für die Anwendung ist die Kompatibilität der Berechnungsweise mit den Ausgabeformaten.

Methode	**Ausgabeformat**	**Distanzmetrik**
Local Binary Pattern	1D Vektor mit 50 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
HSV - Farbwinkelextraktion	1D Vektor mit 256 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
Histogram of oriented Gradients	1D Vektor mit 2304 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
Ringpartitions Hashing	1D Vektor mit 40 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
Strukturmerkmalsbasiertes Hashing	1D Vektor mit 24 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
Farbvektorbasiertes Hashing	1D Vektor mit 201 Dezimalwerten	Euklidischer Abstand, Manhattan, Chi-Quadrat-Distanz
HSV Farbwinkelbasiertes DCT-Hashing	128 Bit Binärhash	Hamming Distanz
Texterkennung mittels Tesseract	String	Levenshtein Distanz

Tabelle 5.1: Zuteilung der Methoden zu den Distanzmetriken

Kapitel 6

Evaluation

In diesem Kapitel werden die Ergebnisse der Evaluation der entwickelten Datenpipeline zur automatischen Klassifikation von Trading Card Game Sammelkarten anhand des Pokémon SKS vorgestellt. Die Pipeline umfasst zunächst ein Segmentierungsmodell, welches in Verbindung mit weiteren Schritten der Vorverarbeitung verwendet wird, um die Karten präzise aus dem Hintergrund eines Scanbilds zuzuschneiden. Die Evaluation dieses Vorverarbeitungsschrittes erfolgt in diesem Kapitel in Abschnitt 6.1.

Auf Basis dieser Grundlage wurde ein Vergleich der Merkmalsextraktionsmethode unter Verwendung verschiedener Distanzmetriken durchgeführt, dessen Ergebnisse in diesem Kapitel evaluiert werden. Dies erfolgt in Abschnitt 6.2.

6.1 Evaluation des Vorverarbeitungskonzeptes

In diesem Abschnitt wird die Kombination der verwendeten Vorverarbeitungsmethoden hinsichtlich ihrer Kartenerkennungsgenauigkeit ausgewertet. Dies erfolgt einerseits durch Evaluationsmetriken, sowie Beispieldaten zur Erfassung der Leistungsfähigkeit der implementierten Vorverarbeitung.

Das Yolo Instanzsegmentierungsmodell wurde unter Verwendung der vortrainierten Version 11 über 20 Epochen fein abgestimmt. Zur Evaluation wurde die beste Version des trainierten Modells validiert. Die Ergebnisse wurden mit der Ultralytics-Version 8.3.49 und der PyTorch-Bibliothek (Version 2.5.1) auf der Google Colab Plattform mittels der bereitgestellten Inference Engine von Ultralytics evaluiert.

Die Validierungsergebnisse des YOLOv11-Modells werden in Bezug auf die Genauigkeit der Boxen (Bounding Boxes) und Masken (Segmentierungsmasken) für alle Klassen dargestellt. Es wurden für die Evaluation insgesamt %15 Prozent der Bilder aus dem originalen Datenset verwendet. Somit beliefen sich die Testdaten für die Evaluation auf 32 Testbilder, welche mit Ihren Ground Truth Daten nach Vorhersage verglichen wurden, um die Evaluationsmetriken zu berechnen.

Tabelle 6.1: Segmentierungsergebnisse des Segmentationsmodells

Metrik	Box P	Box R	Mask P	Mask R
Genauigkeit	98%	94.3%	98%	94.3%

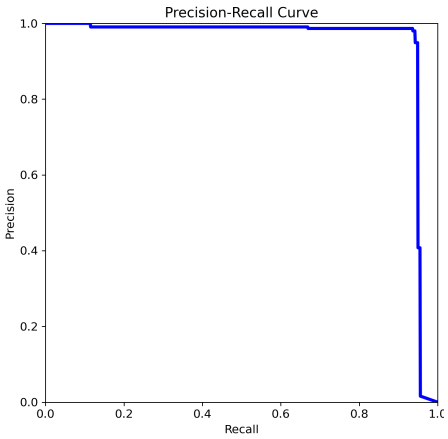


Abbildung 6.1: Bounding Box PR Kurve

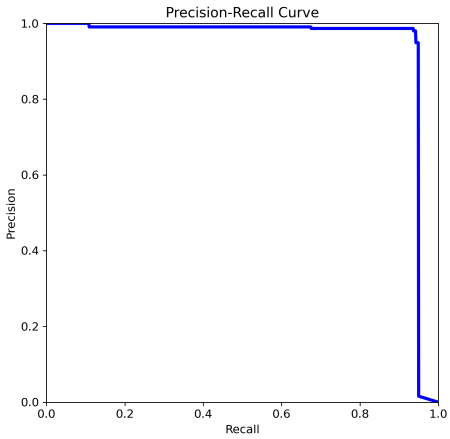


Abbildung 6.2: Segmentierungsmaske PR Kurve

Das Modell erreichte eine Boxvorhersagepräzision von 98%, was bedeutet, dass die vorhergesagten Boxen nahezu perfekt mit den Ground Truth Daten übereinstimmen. Der Box Recall liegt bei 94.3%, was auf nur wenige falsch erkannte Instanzen hinweist. Weitergehend liegt die Präzision bei der Vorhersage von Segmentierungsmasken bei 98 %, was ebenfalls einem nahezu perfekten Match mit den Ground Truth Daten entspricht. Mit einem Recall von 94.3% wurden ebenfalls die meisten Instanzen korrekt segmentiert.

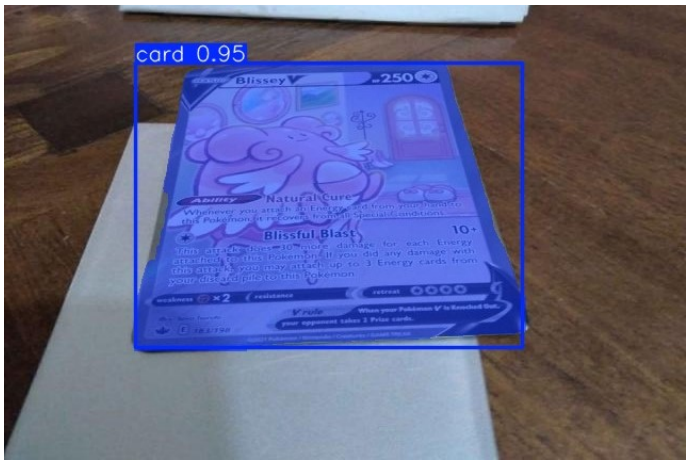


Abbildung 6.3: Evaluation - Geneigtes Bild

Abbildung 6.4: Evaluation - Vogelperspektive

Die Abbildungen 6.4 und 6.3 verdeutlichen die Leistungsfähigkeit des trainierten Modells. Abbildung 6.3 zeigt ein leicht geneigtes Bild mit einer erkannten Bounding Box und einer Segmentierungsmaske in Blau, während Abbildung 6.4 das gleiche für ein Bild aus der Vogelperspektive darstellt. Dies demonstriert, dass das eingesetzte Instanzsegmentierungsmodell nicht nur Bilder zuverlässig verarbeitet, die in optimaler Vogelperspektive aufgenommen wurden, sondern auch bei leicht geneigten und gedrehten Bildern sehr gute Ergebnisse

erzielt. Die Bounding Box und Segmentierungsmasken Präzision p liegt bei den erkannten Instanzen jeweils bei $p > 90\%$

Abbildung 6.5: Validierung – Mehrere Karten aus der Vogelperspektive

Abbildung 6.6: Validierung - Mehrere Karten mit leichter Neigung

Sammelkartenbesitzer lagern Ihre Karten oft in dafür vorgesehenen Ordnern, die jeweils variierende Abstände zwischen den Kartenbehältern aufweisen. Um die Leistungsfähigkeit des Models diesbezüglich zu untersuchen, werden in Abbildung 6.5 und 6.6 solche Sammelordner validiert. Dabei sind die Karten in 6.6 dicht nebeneinander angeordnet, während die Karten in 6.5 eine originale Aufnahme eines solchen Sammelordners zeigen. Zu sehen ist, dass die Karten der Abbildung 6.5 sich trotz hoher Bounding-Box Präzision in der Segmentierungsmaske überlagern. In der Abbildung des Sammelordners ist ein ähnliches Verhalten zu erkennen. Die mögliche Überlappung bei zu dicht anliegenden Karten sollte bei der Aufnahme eines Bildes Beachtung finden. Sammelordner, die etwas mehr Abstand zwischen den Karten ermöglichen, sind zu bevorzugen, da sie die Segmentierungsgenauigkeit verbessern und Überlappungen der Segmentierungsmasken minimieren können. Die erzielten Ergebnisse zeigen, dass das fein abgestimmte Segmentierungsmodell eine hohe Leistungsfähigkeit für die Erkennung von SKS Karten ermöglicht und somit eine solide Grundlage für die Vorverarbeitungsstufe der Pipeline bietet.

6.2 Evaluation der Merkmalsextraktion und Ähnlichkeitssuche

Im Rahmen der Evaluierung der verschiedenen Klassifikationsmethoden, die auf den extrahierten Merkmalen basieren, wurde anstelle einer Präzision-Recall-Kurve die durchschnittliche Klassifikationsgenauigkeit als zentrale Metrik verwendet. Dies liegt an der Art des Testkonzepts, das für diese Evaluierung gewählt wurde.

Im Gegensatz zu probabilistischen Klassifikatoren, die auf Wahrscheinlichkeiten basieren und es ermöglichen, Schwellenwerte für die Klassenzuweisung zu variieren (was zu unterschiedlichen Präzision und Recall-Werten führen kann), gibt es bei diesem Testkonzept keine derartigen Schwellenwertänderungen. Jede Karte wird einer eindeutigen Klasse zugeordnet, und die Vorhersage wird anhand der Seriennummer bestimmt. Dies führt zu konstanten Werten für Präzision und Recall, was die Erstellung einer Präzision-Recall-Kurve unnötig macht.

Die Ergebnisse der durchgeführten Evaluationsszenarien werden für die Auswertung in den Tabellen jeweils für das simple Szenario in Tabelle 6.2, sowie für das komplexe Szenario in Tabelle 6.3 zusammengefasst. Die Tabellen beinhalten die Ergebnisse für die ausgewählten Methoden mit unterschiedlichen Distanzmetriken für die drei Datengrößen. Jede Zeile der Distanzmetriken erfasst die Ergebnisse für die durchschnittliche Klassifikationsgenauigkeit sowie die Bo5-Quote. Für die Auswertung wurden die berechneten Bo5-Quoten durch die Anzahl der Datengrößen dividiert, um einen Durchschnittswert für die Distanzmetriken zu ermitteln. Zusätzlich zu den tabellarischen Darstellungen werden die besten Ergebnisse jeder Methode anhand der durchschnittlichen Distanzmetrik über die drei Datengrößen im Balkendiagramm 6.7 für das einfache Szenario, sowie im Balkendiagramm 6.8 für das komplexe Szenario abgebildet.

Tabelle 6.2: Evaluation der Merkmalsextraktionsmethoden anhand des simplen Szenarios

Methode	Distanzmetrik	Genauigkeit - 50 (%)	Genauigkeit - 500 (%)	Genauigkeit - 5000 (%)	B05-Quote (%)
HOG [Hal13]					
	Euklidische Distanz	12.5	0.35	0	18.93
	Manhattan Distanz	12.5	0.35	0	25.12
	Chi-squared Distanz	12.5	0.35	1.04	26.93
HSV - Hue Extraktion					
	Euklidische Distanz	100	99.64	95.42	99.46
	Manhattan Distanz	100	100	99.59	100.00
	Chi-squared Distanz	100	100	99.27	99.99
LBP [Mül+20]					
	Euklidische Distanz	3.12	0	0	22.28
	Manhattan Distanz	3.12	0	0	22.70
	Chi-squared Distanz	9.37	1.423	0	20.96
Konzentrisches Ringhashing [Tan+16a]					
	Euklidische Distanz	68.75	58.36	33.76	79.31
	Manhattan Distanz	65.62	57.29	33.46	79.99
	Chi-squared Distanz	68.75	60.14	32.36	79.32
Strukturfeature Hash [Qin+16b]					
	Euklidische Distanz	9.37	2.84	0	22.96
	Manhattan Distanz	15.62	1.06	0	21.93
	Chi-squared Distanz	12.5	3.2	0	18.17
Farbvektorbasiertes Hashing [Tan+16a]					
	Euklidische Distanz	56.25	39.14	18.74	74.26
	Manhattan Distanz	56.25	40.56	18.36	73.57
	Chi-squared Distanz	53.12	14.23	0	45.06
HSV - Hue Hashing [Zau10]					
	Hamming	100	80.78	78.89	88.37
Texterkennung					
	Levenstein	89.47	57.65	10.02	54.85

Die Ergebnisse der Pipeline zeigen, dass die Merkmalsextraktionsmethoden deutliche Unterschiede in ihrer Leistung erreichen. Je nach Distanzmetrik und Testdatensatzgröße variieren diese teilweise stark. Besonders hervorzuheben sind die Methoden der HSV-Hue-Extraktion, sowie des HSV-Hue Hashings. Beide erreichen bei der Genauigkeit als auch bei der erreichten B05-Quote konstant exzellente Ergebnisse. Die HSV-Extraktion erreicht über alle Datensatzgrößen nahezu 100% Genauigkeit, wobei alle Distanzmetriken ähnliche Ergebnisse liefern. Lediglich die euklidische Distanz erreichte mit 95.42% bei 5000 Testbildern einen kleinen Abfall im Vergleich mit den anderen Metriken. Die Manhattan Distanz erzielte über alle Datengrößen hinweg besonders gute Leistungen. Besonders eindrucksvoll ist die B05-Quote von 100%, welche mit der Manhattan Distanz erreicht wurde. Das HSV-hue basierte Hashing erzielte ebenfalls gute Leistungen mit 100% Genauigkeit bei 50 Testbildern und knapp 80% bei sowohl 500 und 5000 Testbildern.

Im Mittelfeld der Ergebnisse stehend sind die Methoden der Texterkennung, dem konzentrischen Ringhashing und des Farbvektorbasierten Hashings. Konzentrisches Ringhashing erzielt moderate Ergebnisse über die verschiedenen Datengrößen, jedoch eine konstant hohe Bo5-Quote um die 80%. Das Farbvektorbasierte Hashing hingegen erzielt zwar bei den kleinen Datensätzen 50 und 500 moderate Ergebnisse, fällt jedoch bei größeren Testmengen ab, insbesondere bei der Verwendung der Chi-Quadrat-Distanz. Die Texterkennung mittels der Levenshtein-Distanz erzielt mit etwa 90% Genauigkeit bei 50 Testbildern ein gutes Ergebnis, sinkt jedoch ebenfalls mit Zunahme der Datengröße ab. Bei 500 Bildern liegt die Genauigkeit bereits nur noch bei 57 % und der größte Datensatz von 5000 Testbildern verringert die Genauigkeit auf 10%. Die Bo-5Quote ist mit 54% jedoch moderat.

Besonders schlechte Ergebnisse erzeugten die Methoden HOG, LBP und das Strukturfeature Hashing. HOG erzielt sehr niedrige Genauigkeitswerte von maximal 12,5 % und weist dabei unabhängig von der Distanzmetrik außerdem eine geringe Bo-5Quote auf. Die Methode LBP erreicht bereits bei nur 50 Testbildern lediglich mittels der Chi-Squad-Distanz eine Genauigkeit von knapp 10%. Die Bo5-Quote fällt mit etwa 20% ebenfalls recht gering über alle Distanzmetriken aus. Der Strukturfeature Hash erreicht mit Genauigkeiten zwischen 9% und 16% ebenfalls schlechte Ergebnisse für die kleinste Datengröße von 50. Die Bo5-Quote ist mit 20% ebenfalls im niedrigen Bereich. Die Klassifikationsgenauigkeit des komplexen Szenarios wird anhand des Balkendiagramms 6.7 visualisiert.

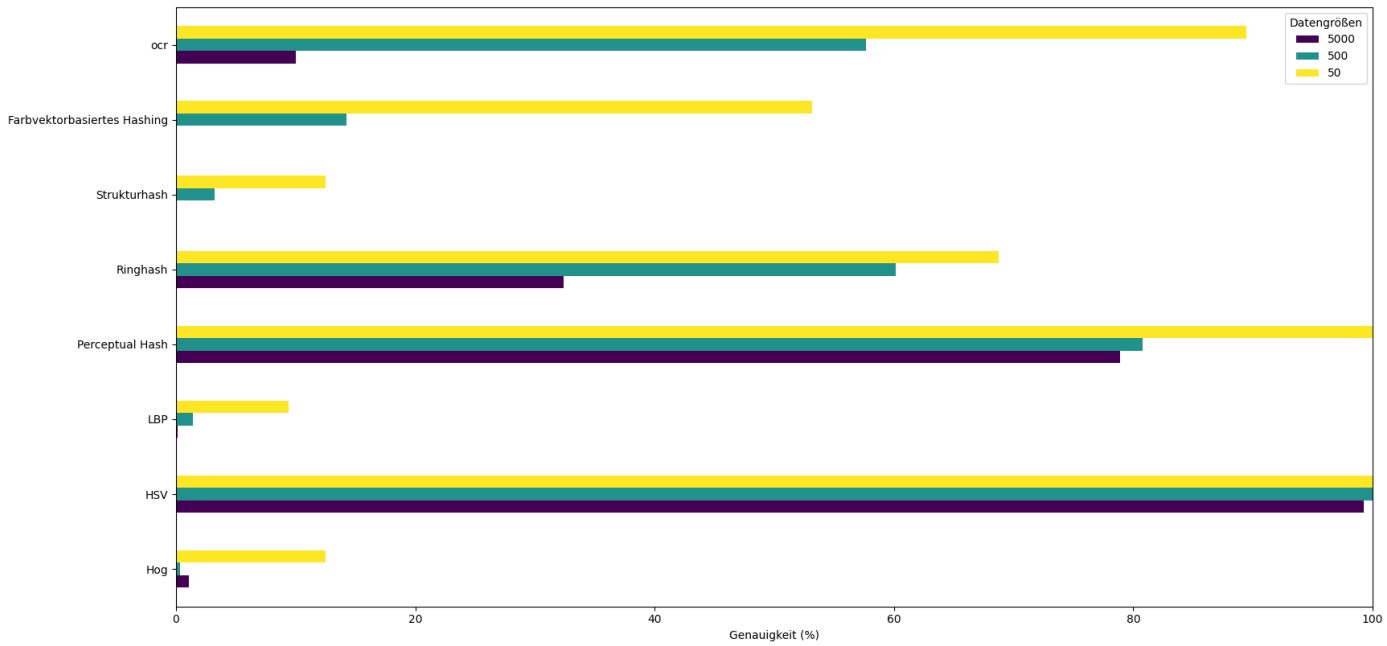


Abbildung 6.7: Visualisierung der Klassifikationsgenauigkeit des simplen Szenarios als Balkendiagramm

Die Auswertung des komplexen Szenarios zeigt signifikante Unterschiede zu den Ergebnissen des simplen Szenarios. Die HSV-Hue basierte Extraktion erzielt wie zuvor eine Genauigkeit von 100% bei der Datengröße

50, fällt jedoch unter Verwendung der Manhattan Distanz mit zunehmender Datengröße von 500 Testbildern auf 62% und bei 5000 auf 45% herab. Die Augmentationen im komplexen Szenarios zeigen hier bereits signifikante Auswirkungen auf die Genauigkeit des Verfahrens. Das helligkeitsbasierte Hashing, welches das zweitbeste Verfahren beim simplen Szenario war, erreicht bereits bei der geringen Datengröße von 50 Testbildern nur die moderate Genauigkeit von 45 %. Die Bo5-Quote ist zwar mit 44% ebenfalls moderat, stellt jedoch einen signifikanten Abfall von der vorherigen Quote des simplen Szenarios dar.

Die Daten der restlichen Methoden fallen sehr gering aus. Lediglich das konzentrische Ringhashing erreicht eine Genauigkeit von über 20% für die kleinste Datengröße bei etwa gleicher Bo5-Quote. Die Bo5-Quoten der Methoden unter 20% Klassifikationsgenauigkeit liegen zwischen 15 und 30 %. Dies macht eine Verwendung dieser Methoden für eine etwaige Best of Five Implementierung aufgrund der niedrigen Quote eher ungeeignet.

Tabelle 6.3: Evaluation der Merkmalsextraktionsmethoden anhand des komplexen Szenarios

Methode	**Distanzmetrik**	**Genauigkeit - 50 (%)**	**Genauigkeit - 500 (%)**	**Genauigkeit - 5000 (%)**	**Bo5-Quote (%)**
HOG [Hal13]					
	Euklidische Distanz	9.09	0.72	0.11	28.99
	Manhattan Distanz	9.09	0.72	0.17	29.47
	Chi-squared Distanz	18.18	0.72	0.17	27.47
HSV - Hue Extraktion					
	Euklidische Distanz	100	36.49	18.04	62.47
	Manhattan Distanz	100	62.04	45.56	83.86
	Chi-squared Distanz	100	51.09	35.63	77.56
LBP [Mül+20]					
	Euklidische Distanz	9.09	0.72	0.35	14.17
	Manhattan Distanz	18.18	0.72	0.35	17.22
	Chi-squared Distanz	9.37	1.42	0.72	17.17
Konzentrisches Ringhashing [Tan+16a]					
	Euklidische Distanz	27.27	2.91	0.87	22.03
	Manhattan Distanz	27.27	2.18	1.10	18.79
	Chi-squared Distanz	18.18	2.18	1.05	19.07
Strukturfeature Hash [Qin+16b]					
	Euklidische Distanz	9.09	2.18	0.29	14.58
	Manhattan Distanz	9.09	2.18	0.11	11.74
	Chi-squared Distanz	0	0.14	0	20.55
Farbvektorbasiertes Hashing [Tan+16a]					
	Euklidische Distanz	18.18	2.91	0.99	27.51
	Manhattan Distanz	18.18	2.91	0	27.51
	Chi-squared Distanz	18.18	0	0	20.79
HSV-Hue Hashing [Zau10]					
	Hamming	45.45	13.13	13.49	44.54
Texterkennung					
	Levenstein	0	0	0	30.49

Die Klassifikationsgenauigkeit des komplexen Szenarios wird anhand des Balkendiagramms 6.8 visualisiert.

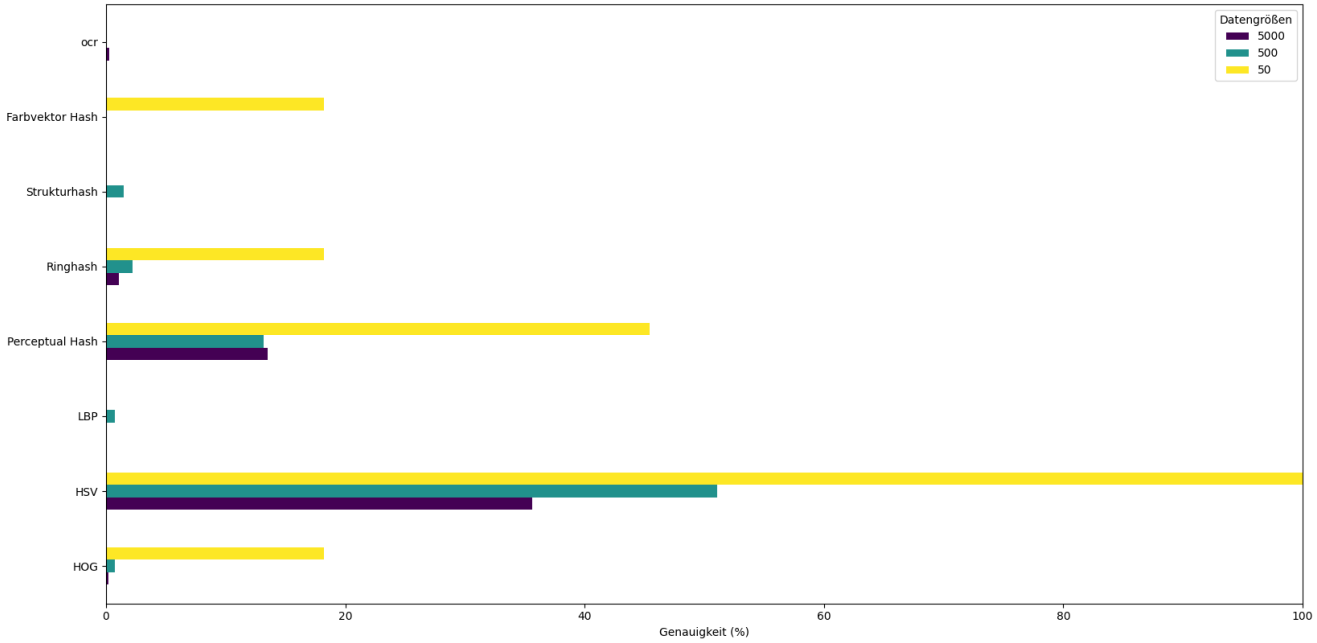


Abbildung 6.8: Visualisierung der Klassifikationsgenauigkeit des komplexen Szenarios als Balkendiagramm

Kapitel 7

Diskussion

In diesem Abschnitt werden die Ergebnisse des Evaluationskapitels diskutiert und Schlussfolgerungen gezogen. Außerdem wird ein Ausblick auf mögliche zukünftige Forschung gegeben.

7.1 Interpretierung der Ergebnisse

Die Segmentierungsergebnisse des Abschnitts 6.1 zeigen, dass das fein abgestimmte Segmentierungsmodell mit einer Segmentierungsgenauigkeit von 98% eine hohe Genauigkeit für die Trennung der SKS Karten von eventuell störenden Hintergründen bietet.

Die Ergebnisse der Evaluation aus 2.6 zeigen, dass die helligkeitsbasierte Merkmalsextraktion sowohl für das simple als auch für das komplexe Szenario die besten Ergebnisse erzielt hat. Für Bilder in optimaler Qualität, wie im Datensatz des simplen Szenarios vorliegend, erreicht die Methode mittels der Manhattan und Chi-Squared Distanzmetrik über alle Datensatzgrößen >= 99 % Genauigkeit. Für das komplexe Szenario erreichte die Manhattan Distanz mit 62 % Genauigkeit für 500 und 45 % für 5000 Testbilder moderate Ergebnisse. Der Abfall der Genauigkeit beim komplexen Szenario ist vermutlich durch den Qualitätsverlust der Karten durch die Perspektivtransformation, sowie Skalierungen durch die Vorverarbeitung bedingt.

Bezüglich der helligkeitsbasierten Merkmalsextraktion sollte beachtet werden, dass der erzeugte Merkmalsvektor 180 Kanäle für die Speicherung der Helligkeitsstufen vorsieht. Diese Dimensionalität erzeugt einerseits einen recht hohen Speicherbedarf, sowie längere Abrufzeiten für den Bildvergleich.

Ein alternatives Verfahren wäre das helligkeitsbasierte Hashing [Zau10], welches ebenfalls besonders für das simple Szenario gute Ergebnisse erzielt hat. Es erweitert die helligkeitsbasierte Merkmalsextraktion mit dem Erzeugen einer Signatur, welche in der hier betrachteten Implementation einen 128 Bit Ausgaberaum aufweist. Die Speichernutzung auf beispielsweise mobilen Endgeräten wäre durch einen so kleinen Merkmalsvektor kaum problematisch. Außerdem verringert sich die Abrufzeit durch den Bildvergleich aufgrund des

binären Ausgaberaums erheblich, da lediglich die 128 Bit des Merkmalsvektors mittels Hamming-Distanz verglichen werden muss. Die Ergebnisse haben besonders gezeigt, dass die Methoden: Strukturhashing, Farbvektorhashing, Ringhashing, welche laut Ihren Autoren besonders resistent gegenüber Störeinflüssen sind, zwar bei kleinen Datengrößen eine hohe bis moderate Genauigkeit liefern, jedoch mit zunehmender Datenanzahl signifikant an Genauigkeit abnehmen. Die Robustheit gegen etwaige Störeinflüsse, wie durch Skalierungseffekte und Rotationsinvarianz, wurde besonders anhand der Ergebnisse des komplexen Szenarios nicht bestätigt, da die Genauigkeit der Methoden mit zunehmender Datengröße stark nachließ.

Bezüglich der Forschungsfrage:„**Welche Kombination aus Bildvorverarbeitungsmethoden und Klassifikationsalgorithmen innerhalb einer Datenpipeline ermöglicht die genaueste eindeutige Klassifizierung von Pokémon Sammelkarten unter verschiedenen Störeinflüssen auf das Eingabebild**“, ist auf Grundlage der erzeugten Ergebnisse die Verwendung eines fein abgestimmten Segmentierungsmodells für die Vorverarbeitung und die helligkeitsbasierte Merkmalsextraktion unter Verwendung der Manhattan Distanz zu empfehlen.

Der Vollständigkeit halber ist zu erwähnen, dass die hier durchgeführte Evaluation der Merkmalsextraktionsmethoden teilweise auf eigenen Implementationen beruht und lediglich ein Pipelinedurchlauf durchgeführt wurde. Besonders das für die Vorverarbeitung verwendete YOLO Instanzsegmentierungsmodell ist nicht konstant in seinem Output. Jeder Durchlauf kann daher die Ergebnisse leicht verändern.

7.2 Zukünftige Forschung

Die in dieser Arbeit entwickelte Pipeline kann erweitert werden, um weitere Methoden für die Merkmalsextraktion miteinander zu vergleichen. Die modulare Aufbauweise ermöglicht außerdem die Ausweitung der Analyse auf andere Kartenspiele. Um die Erweiterung des hier implementierten Konzepts zu ermöglichen, wird die Datenbasis der Arbeit als Anhang beigefügt.

Wie in Abschnitt 7.1 bereits angeschnitten, sind Speicherlösungen im Forschungsbereich des Bildabrufs ebenfalls von hoher Relevanz. Zukünftige Forschung könnte sich daher mit spezifischen optimalen Speicherlösungen, wie Baumstrukturen, für die Merkmalsextraktionsmethoden beschäftigen, um einerseits die Speichernutzung zu optimieren sowie die Abrufzeiten zu minimieren.

Die implementierten Verfahren dieser Arbeit wurden mit einer konstanten Anzahl an Kanälen (Bins) je Methode implementiert. Um die Auswirkung der Variierung der Kanalmengen zu erfassen, könnten weitere Arbeiten diese in mehrere Größenordnungen unterteilen. Als gutes Beispiel könnte dafür das Paper von [LY13] dienen, in dem die Forscher eine solche Unterteilung für Methoden im RGB und HSV-Farbraum durchgeführt haben.

In der Literaturrecherche dieser Arbeit wurde zusätzlich zu den hier behandelten Verfahren der Forschungsbereich des Deep Hashings erwähnt. Dieser verspricht immense Verbesserungen der Genauigkeit bei immer größeren Datensätzen. Zukünftige Arbeiten sollten diese Verfahren in den Methodenvergleich einbeziehen und analysieren, ob diese Methoden wirklich bessere Ergebnisse erzielen.

Abbildungen

Literatur

[AA21] Charles Adjetey und Kofi Adu-Manu. „Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm“. In: *International Journal of Advanced Computer Science and Applications* 12 (Jan. 2021). DOI: 10.14569/IJACSA.2021.0120776 (siehe S. 27, 37).

[Aut24] Wargamer website Authors. *Pokemon card count estimate*. https://www.wargamer.com/pokemon-trading-card-game/how-many-pokemon-cards-arethere/. Accessed: 21.10.2024. 2024 (siehe S. 5).

[BGK15] Dibya Jyoti Bora, Anil Kumar Gupta und Fayaz Ahmad Khan. „Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation“. In: *CoRR* abs/1506.01472 (2015). arXiv: 1506.01472. URL: http://arxiv.org/abs/1506.01472 (siehe S. 5).

[BI88] Manfred Brunner und Werner Ittner. „VIPER: a general-purpose digital image-processing system applied to video microscopy“. In: *Computer Methods and Programs in Biomedicine* 26.2 (1988), S. 167–181. ISSN: 0169-2607. DOI: https://doi.org/10.1016/0169-2607(88)90042-9. URL: https://www.sciencedirect.com/science/article/pii/0169260788900429 (siehe S. 31).

[BM21] R. Biswas und Pablo Medina. „State of the Art: Image Hashing“. In: (Aug. 2021). DOI: 10.48550/arXiv.2108.11794 (siehe S. 35).

[Bri10] Joost van Dreunenm Matthew Wang Brian david Marshall. „Trading Card Game Industry“. In: (Juli 2010), S. 11–12. DOI: 10.1109/ACCESS.2023.3259104 (siehe S. 1).

[bul24a] bulbapedia bulbagarden. *List of Pokémon Trading Card Game expansions*. https://www.citedrive.com/overleaf/. Accessed: 21.10.2024. 2024 (siehe S. 5).

[bul24b] bulbapedia bulbagarden. *List of Pokémon Trading Card Games*. https://bulbapedia.bulbagarden.net/wiki/Pokemon_Trading_Card_Game/. Accessed: 21.10.2024. 2024 (siehe S. 5).

[Bur19] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN: 9781999579517. URL: https://books.google.de/books?id=0jbxwQEACAAJ (siehe S. 28).

[Can86a] John Canny. „A Computational Approach To Edge Detection“. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (Dez. 1986), S. 679–698. DOI: 10.1109/TPAMI.1986.4767851 (siehe S. 17, 32).

[Can86b] John Canny. „A Computational Approach to Edge Detection“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), S. 679–698. DOI: 10.1109/TPAMI.1986.4767851 (siehe S. 17).

[Car+21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski und Armand Joulin. „Emerging Properties in Self-Supervised Vision Transformers“. In: *CoRR* abs/2104.14294 (2021). arXiv: 2104.14294. URL: https://arxiv.org/abs/2104.14294 (siehe S. 33).

[Car+99] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein und Jitendra Malik. *Blobworld: a System for Region-Based Image Indexing and Retrieval (long version)*. Techn. Ber. UCB/CSD-99-1041. 1999. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/5567.html (siehe S. 30).

[Den+10] Jia Deng, Alexander Berg, Kai Li und Fei-Fei Li. „What Does Classifying More Than 10, 000 Image Categories Tell Us?“ In: Bd. 5. Dez. 2010, S. 71–84. ISBN: 978-3-642-15554-3. DOI: 10.1007/978-3-642-15555-0_6 (siehe S. 34, 35, 37).

[DKN08] Thomas Deselaers, Daniel Keysers und Hermann Ney. „Features for image retrieval: An experimental comparison“. In: *Inf. Retr.* 11 (Apr. 2008), S. 77–107. DOI: 10.1007/s10791-007-9039-3 (siehe S. 35, 37, 62).

[DMY15] Reza Davarzani, Saeed Mozaffari und Khashayar Yaghmaie. „Perceptual image hashing using center-symmetric local binary patterns“. In: *Multimedia Tools and Applications* 75 (März 2015). DOI: 10.1007/s11042-015-2496-6 (siehe S. 35–37).

[Dos+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit und Neil Houlsby. „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale“. In: (Okt. 2020). DOI: 10.48550/arXiv.2010.11929 (siehe S. 33).

[DT05] Navneet Dalal und Bill Triggs. „Histograms of oriented gradients for human detection“. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* 1 (2005), 886–893 vol. 1. URL: https://api.semanticscholar.org/CorpusID:206590483 (siehe S. 21).

[Esc+18] Dennis Eschweiler, Thiago Vallin Spina, Rohan C. Choudhury, Elliot Meyerowitz, Alexandre Cunha und Johannes Stegmaier. „CNN-based Preprocessing to Optimize Watershed-based Cell Segmentation in 3D Confocal Microscopy Images“. In: *CoRR* abs/1810.06933 (2018). arXiv: 1810.06933. URL: http://arxiv.org/abs/1810.06933 (siehe S. 32).

[Fan+17] Wen Fang, Hai-Miao Hu, Zihao Hu, Shengcai Liao und Bo Li. „Perceptual Hash-based Feature Description for Person Re-identification“. In: *Neurocomputing* 272 (Juli 2017). DOI: 10.1016/j.neucom.2017.07.019 (siehe S. 36, 37).

[Far21] Hany Farid. „An Overview of Perceptual Hashing“. In: *Journal of Online Trust and Safety* 1.1 (Okt. 2021). DOI: 10.54501/jots.v1i1.24. URL: https://tsjournal.org/index.php/jots/article/view/24 (siehe S. 23, 24).

[GKK18] Afshin Gholamy, Vladik Kreinovich und Olga Kosheleva. „Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation“. In: 2018. URL: https://api.semanticscholar.org/CorpusID:7467506 (siehe S. 56).

[GW92] R.C. Gonzalez und R.E. Woods. *Digital Image Processing*. Addison-Wesley world student series. Addison-Wesley, 1992. ISBN: 9780201508031. URL: https://books.google.de/books?id=C_FRAAAAMAAJ (siehe S. 9).

[Hal13] Kavitha Halappa. „Image Retrieval using HOG and Edge features“. In: Nov. 2013 (siehe S. 34, 37, 60, 74, 76).

[Har+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke und Travis E. Oliphant. „Array programming with NumPy“. In: *Nature* 585.7825 (Sep. 2020), S. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2 (siehe S. 54).

[HB20] Abdul Mueed Hafiz und Ghulam Mohiuddin Bhat. „A Survey on Instance Segmentation: State of the art“. In: *CoRR* abs/2007.00047 (2020). arXiv: 2007.00047. URL: https://arxiv.org/abs/2007.00047 (siehe S. 32).

[HPS09] Marko Heikkilä, Matti Pietikäinen und Cordelia Schmid. „Description of interest regions with local binary patterns“. In: *Pattern Recognition* 42.3 (2009), S. 425–436. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2008.08.014. URL: https://www.sciencedirect.com/science/article/pii/S0031320308003282 (siehe S. 24, 25).

[HS97] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-term Memory“. In: *Neural computation* 9 (Dez. 1997), S. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (siehe S. 25, 26).

[Hun07] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), S. 90–95. DOI: 10.1109/MCSE.2007.55 (siehe S. 55).

[HZ03] Richard Hartley und Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (siehe S. 13, 14).

[Its15] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015 (siehe S. 54).

[Jin+19] Sheng Jin, Hongxun Yao, Xiaoshuai Sun und Shangchen Zhou. „Unsupervised semantic deep hashing“. In: *Neurocomputing* 351 (2019), S. 19–25. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.01.020. URL: https://www.sciencedirect.com/science/article/pii/S0925231219300323 (siehe S. 36).

[JQC23] Glenn Jocher, Jing Qiu und Ayush Chaurasia. *Ultralytics YOLO*. Version 8.0.0. Ultralytics, 10. Jan. 2023. URL: https://ultralytics.com (siehe S. 54).

[Kay07] Anthony Kay. „Tesseract: an open-source optical character recognition engine“. In: *Linux J.* 2007.159 (Juli 2007), S. 2. ISSN: 1075-3583 (siehe S. 54).

[KCB03] M. Kokare, B.N. Chatterji und P.K. Biswas. „Comparison of similarity metrics for texture image retrieval“. In: *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*. Bd. 2. 2003, 571–575 Vol.2. DOI: 10.1109/TENCON.2003.1273228 (siehe S. 38).

[KH24] Rahima Khanam und Muhammad Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. arXiv: 2410.17725 [cs.CV]. URL: https://arxiv.org/abs/2410.17725 (siehe S. 32).

[Ko+19] ByungSoo Ko, Minchul Shin, Geonmo Gu, HeeJae Jun, Tae Kwan Lee und Youngjoon Kim. „A Benchmark on Tricks for Large-scale Image Retrieval“. In: *CoRR* abs/1907.11854 (2019). arXiv: 1907.11854. URL: http://arxiv.org/abs/1907.11854 (siehe S. 32).

[Kra17] Dr Neal Krawetz. *ImageHash*. Version v1.3.6. Accessed: 18.12.2024. 2017. URL: v (siehe S. 54, 63).

[Li+23] Junnan Li, Dongxu Li, Silvio Savarese und Steven Hoi. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. Jan. 2023. DOI: 10.48550/arXiv.2301.12597 (siehe S. 33).

[LY13] Guang-Hai Liu und Jing-Yu Yang. „Content-based image retrieval using color difference histogram“. In: *Pattern Recognition* 46.1 (2013), S. 188–198. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0031320312002713 (siehe S. 79).

[MAJ13] Anand Mishra, Karteek Alahari und CV Jawahar. „Image retrieval using textual cues“. In: *Proceedings of the IEEE international conference on computer vision*. 2013, S. 3040–3047 (siehe S. 33, 37).

[Mat23] Minoru Matsui. „Are all Pokémons created equal? Assessing the value-neutrality of Pokémon TCG design process“. In: *International Association of Societies of Design Research ongress 2024*. 2023 (siehe S. 39, 40).

[mod19] Titancard moderator. *Pokemon card count estimate*. https://titancards.co.uk/blogs/news/a-guide-to-collecting-understanding-pokemon-tcg/. Accessed: 21.10.2024. 2019 (siehe S. 5, 6).

[MP16] MS Meharban und S Priya. „A review on image retrieval techniques“. In: *Bonfring International Journal of Advances in Image Processing* 6.2 (2016), S. 7 (siehe S. 3, 38).

[MR09] Phillip A. Mlsna und Jeffrey J. Rodríguez. „Chapter 19 - Gradient and Laplacian Edge Detection“. In: *The Essential Guide to Image Processing*. Hrsg. von Al Bovik. Boston: Academic Press, 2009, S. 495–524. ISBN: 978-0-12-374457-9. DOI: https://doi.org/10.1016/B978-0-12-374457-9.00019-6. URL: https://www.sciencedirect.com/science/article/pii/B9780123744579000196 (siehe S. 9, 11).

[MRT11] Paulo Martins, Luís Paulo Reis und Luís Teófilo. „Poker Vision: Playing Cards and Chips Identification Based on Image Processing“. In: *Pattern Recognition and Image Analysis*. Hrsg. von Jordi Vitrià, João Miguel Sanches und Mario Hernández. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 436–443. ISBN: 978-3-642-21257-4 (siehe S. 32).

[Mül+20] Martin Müller, Dominik Britz, Laura Ulrich, Thorsten Staudt und Frank Mücklich. „Classification of Bainitic Structures Using Textural Parameters and Machine Learning Techniques“. In: *Metals* 10 (Mai 2020), S. 630. DOI: 10.3390/met10050630 (siehe S. 24, 25, 74, 76).

[OIA19] JS Owotogbe, TS Ibiyemi und BA Adu. „A comprehensive review on various types of noise in image processing“. In: *International Journal of Scientific & Engineering Research* 10.11 (2019), S. 388–393 (siehe S. 8).

[Oqu+24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin und Piotr Bojanowski. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193 (siehe S. 33).

[Pin+13] Dong Ping Tian u. a. „A review on image feature extraction and representation techniques“. In: *International Journal of Multimedia and Ubiquitous Engineering* 8.4 (2013), S. 385–396 (siehe S. 21, 22).

[PT12] Sanjay Patil und Sanjay Talbar. „Content Based Image Retrieval Using Various Distance Metrics“. In: *Data Engineering and Management*. Hrsg. von Rajkumar Kannan und Frederic Andres. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 154–161. ISBN: 978-3-642-27872-3 (siehe S. 26, 67).

[Qin+16a] Chuan Qin, Xueqin Chen, Jing Dong und Xinpeng Zhang. „Perceptual image hashing with selective sampling for salient structure features“. In: *Displays* 45 (2016), S. 26–37. ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2016.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0141938216301020 (siehe S. 35–37, 66).

[Qin+16b] Chuan Qin, Xueqin Chen, Jing Dong und Xinpeng Zhang. „Perceptual image hashing with selective sampling for salient structure features“. In: *Displays* 45 (2016), S. 26–37 (siehe S. 74, 76).

[Red+16] Joseph Redmon, Santosh Divvala, Ross Girshick und Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640 (siehe S. 18–20).

[Ron+14] Weibin Rong, Zhanjing Li, Wei Zhang und Lining Sun. „An improved Canny edge detection algorithm“. In: *2014 IEEE International Conference on Mechatronics and Automation*. 2014, S. 577–582. DOI: 10.1109/ICMA.2014.6885761 (siehe S. 9, 17).

[RTC19] Filip Radenović, Giorgos Tolias und Ondřej Chum. „Fine-Tuning CNN Image Retrieval with No Human Annotation“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.7 (2019), S. 1655–1668. DOI: 10.1109/TPAMI.2018.2846566 (siehe S. 33).

[Rub+20] Biswas Rubel, González-Castro Víctor, Fidalgo Eduardo und Alegre Enrique. „Perceptual image hashing based on frequency dominant neighborhood structure applied to tor domains recognition“. In: *Neurocomputing* 383 (2020), S. 24–38 (siehe S. 35).

[Sac17] Jonathan Sachs. *White paper: Digital Image Basics*. Techn. Ber. M01. Digital Light Color, März 2017, S. 14 (siehe S. 5).

[SP11] Jorge Sánchez und Florent Perronnin. „High-dimensional signature compression for large-scale image classification“. In: Juni 2011, S. 1665–1672. DOI: 10.1109/CVPR.2011.5995504 (siehe S. 34, 37).

[SS18] A. Soycan und Metin Soycan. „Perspective correction of building facade images for architectural applications“. In: *Engineering Science and Technology, an International Journal* 22 (Dez. 2018). DOI: 10.1016/j.jestch.2018.12.012 (siehe S. 14).

[SSB01] Sven Siggelkow, Marc Schael und Hans Burkhardt. „SIMBA — Search Images by Appearance“. In: Sep. 2001, S. 9–16. ISBN: 3540425969. DOI: 10.1007/3-540-45404-7_2 (siehe S. 31).

[Sun+22] Rui Sun, Tao Lei, Qi Chen, Zexuan Wang, Xiaogang Du, Weiqiang Zhao und Asoke K. Nandi. „Survey of Image Edge Detection“. In: *Frontiers in Signal Processing* 2 (2022). ISSN: 2673-8198. DOI: 10.3389/frsip.2022.826967. URL: https://www.frontiersin.org/journals/signal-processing/articles/10.3389/frsip.2022.826967 (siehe S. 28).

[Sur+03] Shamik Sural, Color Histogram, Euclidean Distance, Manhattan Distance, Vector Cosine, Angle Distance, Histogram Intersection, Vadivel Ayyasamy und Arun Majumdar. „Performance comparison of distance metrics in content-based image retrieval applications“. In: *Proc. of Internat. Conf. on Information Technology, Bhubaneswar, India* (Jan. 2003), S. 159–164 (siehe S. 26, 38).

[Tan+16a] Zhenjun Tang, Liyan Huang, Xianquan Zhang und Huan Lao. „Robust image hashing based on color vector angle and Canny operator“. In: *AEU - International Journal of Electronics and Communications* 70 (2016), S. 833–841 (siehe S. 63, 74, 76).

[Tan+16b] Zhenjun Tang, Xianquan Zhang, Xianxian Li und Shichao Zhang. „Robust Image Hashing With Ring Partition and Invariant Vector Distance“. In: *IEEE Transactions on Information Forensics and Security* 11.1 (2016), S. 200–214. DOI: 10.1109/TIFS.2015.2485163 (siehe S. 36, 37).

[Tan+18] Zhenjun Tang, Xuelong Li, Xianquan Zhang, Shichao Zhang und Yumin Dai. „Image hashing with color vector angle“. In: *Neurocomputing* 308 (2018), S. 147–158. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.04.057. URL: https://www.sciencedirect.com/science/article/pii/S0925231218304971 (siehe S. 12, 35–37, 65).

[Tan06] Lip Tan. „Image file formats“. In: *Biomedical imaging and intervention journal* 2 (Jan. 2006), e6. DOI: 10.2349/biij.2.1.e6 (siehe S. 5).

[tea20] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134 (siehe S. 54).

[TSJ15] Giorgos Tolias, Ronan Sicre und Hervé Jégou. „Particular object retrieval with integral max-pooling of CNN activations“. In: (Nov. 2015). DOI: 10.48550/arXiv.1511.05879 (siehe S. 33).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser und Illia Polosukhin. „Attention Is All You Need“. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (siehe S. 33).

[Xia+14] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu und Shuicheng Yan. „Supervised Hashing for Image Retrieval via Image Representation Learning“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 28.1 (Juni 2014). DOI: 10.1609/aaai.v28i1.8952. URL: https://ojs.aaai.org/index.php/AAAI/article/view/8952 (siehe S. 36).

[Xia+21] Zhang Xiaobo, Peng Jinye, Liu Tian und An Zhigang. „Image Retrieval Method Based on Improved Local Binary Pattern“. In: *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. 2021, S. 256–260. DOI: 10.1109/CISCE52179.2021.9445918 (siehe S. 34, 35, 37, 61).

[Yan+23] Suorong Yang, Weikang Xiao, Mengchen Zhang, Suhan Guo, Jian Zhao und Furao Shen. *Image Data Augmentation for Deep Learning: A Survey*. 2023. arXiv: 2204.08610 [cs.CV]. URL: https://arxiv.org/abs/2204.08610 (siehe S. 32).

[You+04] Ian Young, Jan Gerbrands, Lucas Van Vliet, Cip-data Bibliotheek, Den Haag, Young Theodore, Gerbrands Jacob, Van Vliet und Lucas Jozef. „Fundamentals Of Image Processing“. In: (Juli 2004) (siehe S. 15, 16).

[Zau10] Christoph Zauner. „Implementation and Benchmarking of Perceptual Image Hash Functions“. In: 2010. URL: https://api.semanticscholar.org/CorpusID:17075066 (siehe S. 35–37, 48, 62, 63, 74, 76, 78).

[ZTB20] Ebin Zacharias, Martin Teuchler und Bénédicte Bernier. „Image Processing Based Scene-Text Detection and Recognition with Tesseract“. In: *CoRR* abs/2004.08079 (2020). arXiv: 2004.08079. URL: https://arxiv.org/abs/2004.08079 (siehe S. 25).

Akronyme

Bas Bildabrufsystem. 3, 4

Bo5-Quote Durchschnittliche Trefferquote in Fünf Versuchen. 52, 53, 73

CBIR Content based image retrieval. 30, 31

HOG Histogram of oriented Gradients. 21, 22, 80

LBP Local Binary Pattern. 24

LSTM Long Short Term Memory. 25, 26

OCR Optical Character Recognition. 25

SKS Sammelkartenspiel. 2, 3, 5, 6, 39, 70, 72, 78

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht und ist in ähnlicher oder gleicherweise noch nicht als Prüfungsleistung zur Anerkennung oder Bewertung vorgelegt worden.

Rostock, den 14. März 2025