technische universität
dortmund

Klassifizierung von Ziffern in natürlichen Szenen

Jan-Philipp Tarnowski
5. März 2018

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Eugen Rusakov, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

INHALTSVERZEICHNIS

1 EINLEITUNG

Die maschinelle Erkennung von Text in Bildern ist ein schwieriges Problem, welches viele interessante Anwendungen in der Industrie bietet. In den Straßen heutiger Städte kommen unzählige Straßenschilder, Werbeplakate und Gebäudefassaden von Unternehmen vor, die Text und Zahlen beinhalten. Durch den kontinuierlichen Fortschritt in der Informationstechnik existieren heute eingebettete Systeme und Handys, die mit leistungsfähigen Prozessoren und hochauflösenden Kameras ausgestattet sind. Diese Geräte sind in der Lage, in Alltagssituationen Bilder aufzunehmen, die visuellen Text, wie den Schriftzug eines Restaurants, oder Hausnummern enthalten. Dieser *Szenentext* enthält häufig wichtige Informationen über den Inhalt des Bildes. Abbildung 1.0.1 zeigt verschiedene Beispiele für Szenentext. Durch die maschinelle Analyse der Szenentextelemente eröffnen sich neue Möglichkeiten für die Gewinnung semantischer Informationen aus Fotos.

Die automatische Erkennung von Text- und Zahlenelementen in Fotos birgt vielseitige Anwendungen für die Automatisierung der Industrie sowie für Verbraucher: Selbstfahrende Fahrzeuge können Bildinformationen aus ihrer Umgebung sammeln und erkannte Szenentextelemente zur Navigation nutzen, sowie aus diesen semantische Informationen gewinnen [PCN10]. Blinde und sehbehinderte Menschen können für sie unkenntlichen Text abfotografieren und von einem Computer erkennen lassen (vgl. [YD15]).

Ein wichtiger Unterbereich der Erkennung von Szenentext ist das automatische Auslesen von Hausnummern. Zusätzlich zu den bereits aufgeführten Anwendungszwecken ist die Analyse von Hausnummern sehr nützlich für die Verbesserung von Karten und Navigationsdiensten: Für Dienste wie Google Street View werden enorme Mengen an Fotos von Hausfassaden erstellt, aus welchen automatisch Hausnummern erfasst werden. Diese Ziffern sind mit ihren Koordinaten und ihrer geografischen Ausrichtung versehen, wodurch Kartendienste automatisch um Hausnummern ergänzt werden können (vgl. [NWC+11]).

Im Gegensatz zu anderen Formen der Texterkennung, etwa für maschinengedruckte Dokumente, stellt Szenentext ein besonders schwieriges Problem dar, welches viele eigene Herausforderungen beinhaltet: Text in Fotos ist für einen Computer strukturell nicht immer eindeutig von anderen Objekten im Bild zu unterscheiden. Blätter, Zäune, oder Fenster können visuell ähnliche Eigenschaften aufweisen, und daher mit

tatsächlichen Textelementen verwechselt werden (siehe [YD15]). Während maschinengedruckte Dokumente meistens aus nur einer Schriftart bestehen, zeichnet sich Szenentext durch eine enorme Vielfalt in seinen visuellen Eigenschaften aus. Häufig kommen auf Schildern stilisierte Textelemente vor, die maschinell schwierig einzuordnen sind. Zusätzlich können beim Fotografieren der Bilder unter ungünstigen Bedingungen Verfälschungen in das Bild geraten, welche die Erkennung zusätzlich erschweren: Häufig werden Hausnummern aus einem Winkel fotografiert, wodurch die Bilder der Zahlen perspektivisch verzerrt werden. Bei schlechten Lichtverhältnissen kann der Kontrast des Fotos gemindert werden, und Rauschen in das Bild geraten. Wackelt die Kamera beim Erstellen des Fotos oder wird die Zahl aus großer Entfernung fotografiert, kann das Bild unscharf werden, wodurch einzelne Details der Hausnummer verschwinden. Weiterhin können Schatten, Blendungen oder andere Fremdobjekte im Bild Teile der Zahl verdecken, wodurch die visuelle Struktur der Zahl verfälscht wird [YD15, ZYB16].

Abbildung 1.0.1: Von Gebäudefassaden und Schildern erfasste Szenentextelemente. Gut erkennbar sind die unterschiedlichen Schriftarten, häufig werden auf dem selben Schild mehrere Typen verwendet. Stilistische Elemente wie Rahmen und stilisierte Buchstaben können zusätzliche Schwierigkeiten bereiten. Bilder gesammelt aus dem Street View Text (SVT) Datensatz, vorgestellt in [WB10].

1.1 RAHMEN DER ARBEIT

1.1.1 *Klassifikation*

Das Thema der Bachelorarbeit ist die automatische Erkennung von Hausnummern. Im Gegensatz zu anderen Arbeiten wie [GBI+13], welche gesamte Hausnummern

analysieren, betrachtet diese Arbeit Hausnummern auf der Ebene einzelner Ziffern. Die indisch-arabischen Ziffern bilden ein Dezimalsystem, welches 10 unterschiedliche *Klassen*, die Ziffern von 0 bis 9, beinhaltet. Es gilt, einen *Klassifikator* zu erzeugen, der mit möglichst hoher Zuverlässigkeit das Bild einer Ziffer ihrer zugehörigen Klasse zuordnet. Zusätzliche Zeichen in den Hausnummern, z. B. Buchstaben, werden hier vernachlässigt.
Zunächst werden aus den Ziffernbildern ausschlaggebende Merkmale (*features*) extrahiert. In der *Merkmalsrepräsentation* des Bildes sollen möglichst nur Elemente des Bildes enthalten sein, die dem Klassifikator eine eindeutige Unterscheidung der verschiedenen Ziffern ermöglichen. Für verschiedene Problembereiche können unterschiedliche Merkmale für die Klassifikation ausschlaggebend sein (vgl. [DHS01] Kapitel 1.2). Im Bereich der Hausnummern kann beispielsweise die Farbe einer Ziffer allgemein vernachlässigt werden, da diese keinerlei Einfluss auf ihre Bedeutung nimmt — viel zweckmäßiger ist die Form der Zahl, welche sich über die Verteilung und Richtung der Kanten im Bild darstellen lässt [GY01]. In dieser Arbeit wird der *Histogram of Oriented Gradients* (HOG) Deskriptor verwendet. Das HOG-Verfahren unterteilt ein Pixelbild in getrennte Zellen und berechnet in jeder dieser Zellen ein Histogramm über die Verteilung der lokalen Bildgradienten. Die Histogramme werden darauf zellenübergreifend normiert und zu einem Vektor konkateniert. Der Klassifikator wird mit einer Menge an Trainingsbildern, welche vorher manuell mit ihrer jeweiligen Ziffer annotiert wurden, *trainiert*. Aus den Trainingsbeispielen soll der Klassifikator die ausschlaggebenden *Muster* der einzelnen Klassen erlernen. Nach dem Training soll der Klassifikator anhand der gelernten Muster auch unbekannte Bilder zuverlässig in ihre zugehörige Klasse einordnen können. Für die Arbeit wurden mehrere Konfigurationen des *K Nearest Neighbor* (KNN) Klassifikators und der *Support Vector Machine* (SVM) evaluiert.

1.1.2 *Augmentation*

Sowohl der K Nearest Neighbor Klassifikator als auch die Support Vector Machine benötigen große Mengen an Trainingsdaten, um bei der Klassifikation genaue Ergebnisse erzielen zu können. Jegliche Trainingsdaten müssen vorher von Hand mit ihrer zugehörigen Klasse annotiert werden, was einen sehr großen manuellen Arbeitsaufwand erfordert (vgl. [CBV09, DT17]). Dieses Problem soll durch den Einsatz von *Augmentationen* gemindert werden. Bilder aus der Trainingsmenge werden hierbei maschinell verändert und der Menge wieder zugeführt. Auf diesem Weg lässt sich mit wenig Aufwand die bestehende Trainingsmenge vervielfachen, und die visuelle Variation

der Bilder innerhalb der Menge steigern. Augmentationen können entweder im *data space* oder im *feature space* durchgeführt werden [WGSM16]. Bei der Augmentation im data space werden die Pixelwerte direkt bearbeitet, während Augmentationen im *feature space* auf den Merkmalsrepräsentationen der Bilder durchgeführt wird [DT17]. Im data space existieren *geometrische* und *fotometrische* Augmentationen [TN17]. Geometrische Verfahren verschieben die Position der Pixel im Bild, wodurch die Form des Bildes verändert wird. Beispiele für geometrische Verfahren sind Cropping, sowie affine und perspektivische Transformationen. Fotometrische Verfahren verändern hingegen den Farbgehalt einzelner Pixel. Mögliche Verfahren sind hier u. a. die Veränderung von Helligkeit und Kontrast des Bildes, sowie Unschärfe- oder Rauschfilter. Es gilt Augmentationsverfahren auszusuchen, die möglichst variierte Trainingsbeispiele erzeugen, ohne dabei die zu erkennende Ziffer so stark zu verfälschen, dass sie unkenntlich wird (vgl. [WGSM16]).
Bei Augmentationen im feature space werden hingegen neue Merkmalsvektoren generiert. Beispiele für Augmentationen im feature space sind das Behaften eines Merkmalsvektors mit einem zufälligen Rauschen [DT17], oder das Generieren neuer Merkmalsvektoren durch die Interpolation aus mehreren existierenden Vektoren [CBHK02]. Der Einsatz von Datenaugmentation kann in verwandten Problembereichen, wie z. B. der Erkennung von handschriftlichen Ziffern, bereits sehr vielversprechende Ergebnisse erzielen [CMGS10]. Im Bereich der Hausnummernerkennung wurden in [GBI+13] und [BMK14] durch kleine Verschiebungen der Zahlenbilder eine subtile Form der Datenaugmentation vollzogen. Für die Arbeit wurden verschiedene Augmentationen im data space evaluiert, welche mit unterschiedlichen Häufigkeiten auf die Trainingsbilder angewandt wurden.

Abbildung 1.1.1: Beispiele für die Datenaugmentierung. Aus vier Ursprungsbildern wurden je drei neue Bilder hergestellt. Die Bilder sind durch affine Transformationen sowie durch Anpassung von Helligkeit und Kontrast verfälscht. Teils sind die Bilder invertiert. Ursprungsbilder aus dem SVHN-Datensatz, vorgestellt in [NWC+11].

1.2 GLIEDERUNG DER ARBEIT

Die Arbeit ist folgendermaßen gegliedert: Kapitel 2 erklärt die verwendeten Methodiken im Detail: Der HOG-Deskriptor, der KNN-Klassifikator und die SVM werden in diesem Kapitel genauer erläutert. In Kapitel 3 werden verwandte Arbeiten aufgeführt, die sich mit ähnlichen Problemen wie diese Arbeit beschäftigen, und diese Arbeit beeinflusst haben. Kapitel 4 beschreibt die angewandte Methodik: Es werden die verwendeten Datensätze für die Trainingsphase und die Evaluation vorgestellt, und es wird der genaue Versuchsaufbau erläutert. Zusätzlich werden in diesem Kapitel die in dieser Arbeit verwendeten Augmentationsverfahren betrachtet. In Kapitel 5 wird eine Evaluation der Methodik vollzogen. Es werden die Versuchsergebnisse vorgestellt, verschiedene Parameterkonfigurationen der Klassifikatoren evaluiert, und erfolgreiche Augmentationsverfahren vorgestellt. Schlussendlich werden in Kapitel 6 die Ergebnisse zusammengefasst, und ein Fazit gezogen.

2 GRUNDLAGEN

Für die weitere Verarbeitung der Bilder und ihrer Merkmalsdeskriptoren ist eine mathematische Betrachtungsweise notwendig. Digitalkameras lösen das in ihren Sensor einfallende Licht in ein regelmäßiges Gitter diskreter Bildpunkte auf, und speichern das Bild anschließend als Matrix ab. Die Elemente dieser Matrix werden *Pixel* genannt. Die Dimensionen der Matrix werden als die *Auflösung* des Bildes bezeichnet. Je höher die Auflösung, desto feinere Details können im Bild erfasst werden. Wird nur die Helligkeit und keine Farbinformation gespeichert, ist für die Repräsentation des Bildes i. d. R. eine Belichtungsmatrix ausreichend. Ein solches Bild wird ein *Graustufenbild* genannt. Farbbilder werden i. d. R. durch drei Farbkanäle für je rot, grün und blaue Farbinformationen dargestellt. Neben diesen *RGB-Bildern* existieren auch andere *Farbräume* wie *HSV* (Helligkeit, Sättigung, Farbe), zwischen denen es Umrechnungsverfahren gibt (siehe [NFHS12], Kapitel 3.5, [Pri15], Kapitel 3.2). Je nach Anwendung kann eine unterschiedliche Repräsentation der Bilddaten gewählt werden. Im Rahmen dieser Arbeit werden alle Bilder als Graustufenbilder oder RGB-Farbbilder dargestellt. Die Pixelwerte werden auf den Wertebereich $[0, 1]$ skaliert. Der Wert 0 entspricht der geringsten Beleuchtung (Schwarz), während der Wert 1 der höchsten Beleuchtung entspricht: Für Graustufenbilder ist dies die Farbe Weiß, während in RGB-Farbbildern dieser Wert der maximalen Farbintensität für den jeweiligen Farbkanal entspricht.
Die Pixelwerte werden digital als 64-Bit-Fließkommazahlen repräsentiert, um numerische Berechnungen im Computer zu ermöglichen. Dieser Datentyp erlaubt eine hohe Präzision in den Berechnungen, schließt kleine Rundungsfehler jedoch nicht aus [Gol91]. Farbbilder können in Graustufenbilder umgewandelt werden, die Farbinformation geht hierbei jedoch verloren. Näheres zu der digitalen Darstellung von Farben, und den technischen Hintergründen verschiedener Farbmodelle ist in [NFHS12], Kapitel 3.5 zu finden.
Um die Klasse unbekannter Hausnummernbilder zu bestimmen wird ein Klassifikator trainiert. Dazu erhält der Klassifikator eine Menge an Trainingsbeispielen, die mit ihrer zugehörigen Klasse annotiert sind. Nach der Trainingsphase ist der Klassifikator in der Lage, für die Klasse eines unbekannten Objektes eine Vorhersage zu treffen.
Aus jedem Trainingsobjekt wird zunächst eine Merkmalsrepräsentation ermittelt. Das Ziel ist es, für ein Objekt so weit wie möglich genau die Merkmale zu finden, die

indikativ für die zu dem Objekt gehörige Klasse sind, und alle anderen Informationen zu verwerfen (siehe [TK08], Kapitel 5). Dazu werden auf einer Bildmatrix die HOG-Deskriptoren berechnet, welche in Abschnitt 2.2 erläutert werden. Das HOG-Verfahren wird häufig für die Erkennung von Objekten in Szenen verwendet [DT05], und kann durch die Variation mehrerer *Hyperparameter* angepasst werden. Das Verfahren erzeugt für jedes Bild einen zugehörigen *Merkmalsvektor*. Diese Merkmalsvektoren dienen als Eingabe für den Klassifikator.
Als Klassifikatoren werden entweder der K Nearest Neighbors Klassifikator oder die Support Vector Machine benutzt. Beide Klassifikatoren interpretieren die Merkmalsvektoren als n-dimensionale Vektoren im euklidischen Raum, zeichnen sich aber durch ihre unterschiedlichen Methoden aus, verschiedene Klassen voneinander zu trennen: Der KNN-Klassifikator bestimmt die Klasse eines unbekannten Datenpunktes anhand der umliegenden Trainingsdaten, während die SVM eine Hyperebene im Raum konstruiert, welche als Trennbarriere zwischen zwei verschiedenen Klassen fungiert. Die Klassifikatoren werden in ihren jeweiligen Abschnitten, Abschnitt 2.3 für den KNN und Abschnitt 2.4 für die SVM näher beschrieben.

2.1 KLASSIFIKATION

Die in der Arbeit vorgestellten Klassifikatoren arbeiten beide nach dem Prinzip des *supervised learnings*. Dies bedeutet, dass für die Trainingsbeispiele die zugehörigen Annotationen vorliegen. Es existiert eine endliche Menge an Klassen $\{K_1, K_2, ..., K_m\}$. Der Klassifikator erhält als Eingabe eine Menge aus n-dimensionalen reellwertigen Merkmalsvektoren, die mit ihrer Klassenzugehörigkeit annotiert sind. Die Trainingsmenge lässt sich wie folgt darstellen ([Mur12], Abschnitt 1.1.1):

$$X_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \tag{2.1.1}$$

Aus den Merkmalsvektoren $\mathbf{x}_i$ geht ein n-dimensionaler Vektorraum, der *Merkmalsraum*, hervor. Ein Klassifikator partitioniert den Merkmalsraum in m räumliche Teilregionen, von denen jede Region zu einer der m Klassen zugeordnet wird ([TK08], Kapitel 2.3). Um ein unbekanntes Sample zu klassifizieren, wird dieses in dieselbe Merkmalsrepräsentation wie die Trainingssamples gebracht. Die Region, in die das Objekt eingestuft wird, bestimmt die Klassenzugehörigkeit des Objekts.

Evaluation

Es ist notwendig, die Leistung eines Klassifikators zu evaluieren, um die Klassifikationsleistung zu optimieren. Der Klassifikator muss möglichst gut *generalisieren*, um unbekannte Daten besser einordnen zu können (vgl. [DHS01], S.7f). Wird der Klassifikator zu spezifisch auf die vorliegenden Trainingsdaten abgestimmt, besteht die Gefahr des *overfittings*, welches es zu vermeiden gilt ([DHS01], Abschnitt 1.3.3, [Mur12], Abschnitt 1.4.7). Häufig sind die Trainingsdaten durch Merkmale verfälscht, die nicht zu den tatsächlichen Eigenschaften der Klasse gehören (*noise*) ([DHS01], Abschnitt 1.3.2). Werden diese Merkmale fälschlicherweise als Bestandteil der Klasse anerkannt, kann dies die Klassifikationsleistung mindern. Es ist daher nicht zwingend erforderlich, dass der trainierte Klassifikator die Trainingsmenge selbst optimal klassifiziert, da stattdessen eine bessere Generalisierung und Robustheit gegen noise gewünscht sind. Ein Beispiel hierfür ist in Abbildung 2.1.1 zu sehen.

Um die Leistung eines Klassifikators zu evaluieren werden u. a. zwei Verfahren häufig verwendet. Einerseits lässt sich die Leistung durch eine gesonderte Testmenge X_{test} prüfen. Die Klassen der Testobjekte werden vom Klassifikator geschätzt, und darauf mit den tatsächlichen, bekannten Klassenlabels verglichen. Die Verwendung einer einzigen, vorher bekannten, Testmenge birgt jedoch erneut die Gefahr des overfittings im Bezug auf die Testmenge.
Alternativ wird zur Evaluation häufig eine *Kreuzvalidierung* (cross-validation) verwendet. Hierzu wird eine Partition der Trainingsmenge X_{train} in f Teilmengen, oder *folds* vorgenommen. Der Klassifikator wird stets mit $f-1$ der Teilmengen trainiert, und auf der verbleibenden Teilmenge evaluiert. Auf diesem Weg entstehen f Klassifikationsergebnisse. Es existieren verschiedene Evaluationsmetriken, auf die in Kapitel 5 eingegangen wird.

2.2 HISTOGRAM OF ORIENTED GRADIENTS

Der Histogram of Oriented Gradients (HOG) Deskriptor wurde 2005 von Dalal und Triggs in [DT05] vorgestellt. Das Ziel des HOG-Deskriptors ist es, genau die Form eines Objektes zu erfassen, und dabei möglichst unanfällig gegen räumliche Beleuchtungsunterschiede im Bild zu sein (siehe [DT05]). Aus diesem Grund werden HOG-Deskriptoren häufig in Mustererkennungsaufgaben eingesetzt, in denen die gesuchten Objekte eine ähnliche, regelmäßige Form haben: Dalal und Triggs nutzen den Deskriptor zur Erkennung von Personen ([DT05], [Sze10] Kapitel 14.1.2), und das Verfahren wurde bereits im Bereich des Szenentextes [WB10, PLH10] und der Hausnummernerkennung [NWC+11] untersucht.

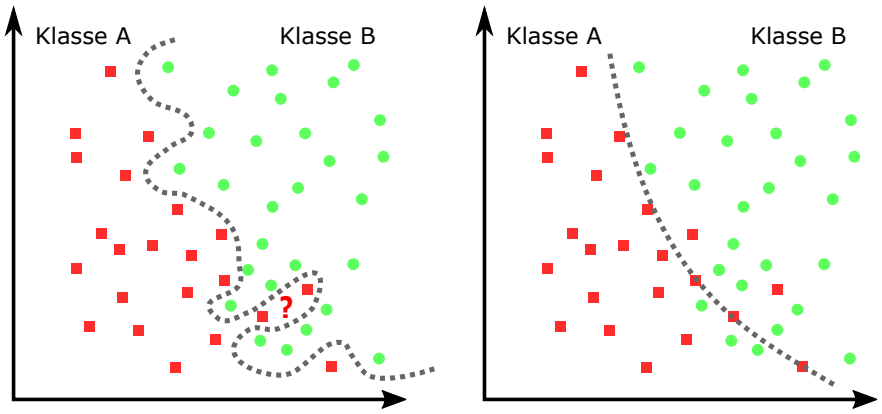


Abbildung 2.1.1: Links wurde zwischen beiden Klassen die Trennlinie so gezogen, dass jedes Trainingssample richtig eingestuft wird. Ein Sample an der Position des Fragezeichens gehört wahrscheinlich zur Klasse B, wird aber in die Klasse A eingestuft. Rechts wird hingegen eine glatte Trennlinie gezogen, durch die einige Trainingssamples falsch klassifiziert werden. Diese Trennlinie entspricht wahrscheinlich dennoch eher der tatsächlichen Verteilung der Daten. Abbildung nach [DHS01], S.6f

HOG-Deskriptoren erfassen in einem Bild die Intensität, Richtung und Verteilung der lokalen Bildgradienten. Dazu wird ein Bild in ein regelmäßiges Gitter aus Zellen unterteilt, in denen ein Histogramm über die Gradienten erstellt wird. Darauf findet eine zellenübergreifende Normierung statt. Das Verfahren kann auf Farb- oder Graustufenbilder angewandt werden und berechnet aus diesen einen Merkmalsvektor. Im Verfahren existieren mehrere frei wählbare Hyperparameter, die einen zum Teil großen Einfluss auf den Ergebnisvektor, und folglich die nachfolgende Klassifikationsleistung nehmen. Dalal und Triggs führen in ihrer Arbeit mehrere Hyperparameter auf und evaluieren in ihren Experimenten den Einfluss verschiedener Werte für die Parameter. Das Verfahren wird im Folgenden beschrieben, wählbare Parameter werden entsprechend gekennzeichnet.

2.2.1 *Berechnen der HOG-Deskriptoren*

Die Berechnung der HOG-Deskriptoren ist ein mehrstufiger Prozess, der sich wie folgt aufbaut. Näheres Details zu einzelnen Implementationsschritten, sowie die Evaluationsergebnisse für einzelne Parameter, sind der Arbeit von Dalal und Triggs

[DT05] zu entnehmen.

Vorbearbeitung

Als erstes kann das Bild optional vorbearbeitet werden. Dalal und Triggs probieren in ihrer Arbeit verschiedene Gammakorrekturen aus, merken aber an, dass diese nur einen geringen Effekt auf die Erkennungsrate haben [DT05].

Gradientenbild berechnen

Als nächstes werden die sich im Bild befindlichen Kanten ermittelt. Dies geschieht durch die Berechnung des *Gradientenbildes*. Das Gradientenbild beschreibt für jedes Pixel des Originalbildes die Magnitude und die Richtung des lokalen Farb- oder Helligkeitsunterschieds. Das Gradientenbild ist durch die partielle Ableitung in x- und y-Richtung wie folgt gegeben (siehe [GW02] S.134):

$$\nabla \mathbf{f} = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \tag{2.2.1}$$

An einer Stelle $G(i,j)$ des Bildes kann die Magnitude des Bildgradienten wie folgt ermittelt werden:

$$G_{mag}(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{2.2.2}$$

Die Ausrichtung des Gradienten berechnet sich folgendermaßen:

$$G_{dir}(i,j) = \tan^{-1}\left(\frac{G_x(i,j)}{G_y(i,j)}\right) \tag{2.2.3}$$

Die Ableitung der Bildmatrizen kann über eine zweidimensionale diskrete *Faltung* approximiert werden. Näheres zur diskreten Signalfaltung im Kontext digitaler Bilder

findet sich in [NFHS12], Kapitel 5.2. Für den Zweck der Kantenerkennung lassen sich verschiedene *Faltungsmasken* (auch: Kerne) für die x und y-Richtung verwenden.

$$L_x \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad L_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{2.2.4}$$

Gebräuchlich sind u. a. Gleichung 2.2.4 [DT05]. L_x wird hier zur Erkennung von vertikalen Gradienten, L_y für horizontale Gradienten verwendet. Weiterhin werden häufig der Sobeloperator ([NFHS12] S.138), der Prewitt-Operator ([GW02] S.578) und der Laplace-Operator ([NFHS12] S.142) verwendet. Demnach schreibt das HOG-Verfahren nicht vor, welche Maske genutzt werden soll. Zusätzlich kann es vor der Kantendetektion sinnvoll sein, das Bild zu glätten, um Rauschen zu entfernen. Glättungen können mit einer ähnlichen Faltungsoperation durchgeführt werden. Näheres zu Glättungsmasken findet sich in [NFHS12], Kapitel 5.3. Bei Farbbildern berechnen Dalal und Triggs für jeden Farbkanal das Gradientenbild und nutzen für jedes Pixel den Wert des Farbgradienten mit der größten Magnitude. Abbildung 2.2.1 zeigt eine grafische Repräsentation der Kantenerkennung. Für die Ermittlung der Gradientenhistogramme im nächsten Schritt ist es erforderlich, sowohl Magnitude wie auch Richtung der Gradienten zu erfassen.

Abbildung 2.2.1: Berechnung der Bildgradienten. Von links nach rechts: Originales Graustufenbild, Vertikales Gradientenbild, Horizontales Gradientenbild, Gradientenmagnituden, Richtungen der Gradienten, repräsentiert durch Richtungspfeile, die an Stellen mit hoher Gradientenintensität auf das Bild gelegt wurden. Die Richtungen sind farbkodiert.

Zellen und Gradientenhistogramme

Die HOG-Merkmalsrepräsentation eines Bildes behält die Position der Bildmerkmale bei. Dies wird erreicht, indem Informationen über die Gradienten innerhalb getrennter Bildbereiche gesammelt werden. Dazu wird das Bild in ein regelmäßiges Gitter aus *Zellen* unterteilt. Zellen können nach [DT05] rechteckig oder radial geformt sein. Die Größe der Zellen ist ein weiterer wählbarer Parameter. Innerhalb jeder Zelle wird nun der lokale Gradient jedes enthaltenen Pixels ermittelt und gemäß seiner Richtung in ein Histogramm eingetragen. Das Histogramm unterscheidet n Richtungen, die jeweils immer im gleichen Winkel zueinander stehen. Enthält das Histogramm beispielsweise 4 *bins*, so repräsentieren diese die Richtungen 0°, 90°, 180° und 270°. Für jedes Pixel der Zelle wird seine jeweilige Gradientenmagnitude in das Histogramm eingetragen. Liegt der Winkel des Gradienten zwischen zwei Werten des Histogramms, so wird der Wert proportional auf die beiden umliegenden *bins* verteilt. Anstelle der Gradientenmagnitude g lässt sich eine Funktion der Magnitude $f(g)$ nutzen.
Die Gradienten können dabei *vorzeichenbehaftet* oder *nicht vorzeichenbehaftet* gewertet werden. Bei der vorzeichenbehafteten Betrachtungsweise werden die Richtungsklassen des Histogramms von 0° bis 180° gespannt. Gradientenwinkel zwischen 180° und 360° werden dabei um 180° reduziert, wodurch diese gleich wie ihre genau umgekehrten Winkel interpretiert werden. Ist das Histogramm nicht vorzeichenbehaftet, werden die *bins* von 0° bis 360° gespannt (vgl. [DT05]).

Blocknormierung

Die bedeutsamste Eigenschaft des HOG-Deskriptors ist die zellenübergreifende Normierung der Gradientenhistogramme. Der Merkmalsvektor wird aus der Konkatenation von *Blöcken* gebildet, die in einem regelmäßigen Gitter über das Bild gelegt werden. Ein Block wird hierbei berechnet, indem die Histogrammvektoren mehrerer angrenzender Zellen konkateniert und anschließend blockweit normiert werden. Einzelne Blöcke können sich überlappen, wodurch Zellen mehrmals in verschiedenen Blöcken vorkommen können. Durch unterschiedliche Normierungsfaktoren innerhalb einzelner Blöcke kann sich der Histogrammvektor einer Zelle dennoch zwischen zwei Blöcken unterscheiden. Die redundanten Zellenvorkommen erhöhen die Größe und den Speicherverbrauch des HOG-Vektors, führen aber in der Praxis zu besseren Erkennungsraten [DT05].
Für die Normierung der Blöcke schlagen Dalal und Triggs vier mögliche Verfahren vor. Ein Blockvektor $\mathbf{v}$ wird normiert durch:

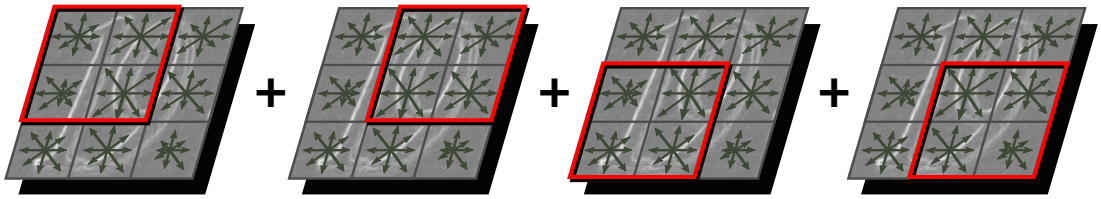


Abbildung 2.2.2: Veranschaulichung der Blocknormierung in R-HOGs. Die rot umrandete Fläche stellt stets einen der Blöcke dar.

- *L2-norm*:

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon^2}} \tag{2.2.5}$$

- *L2-hys*: Nach der L2-norm werden die Werte von $\mathbf{v}$ auf 0,2 beschränkt und erneut normiert.
- *L1-norm*:

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon} \tag{2.2.6}$$

- *L1-sqrt*:

$$\mathbf{v} \rightarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}} \tag{2.2.7}$$

ϵ ist ein Regularisierungsparameter, dessen Auswirkung vernachlässigt werden kann [DT05].

Ähnlich zu den Zellen sind die Maße und die Form der Blöcke frei wählbar. Dalal und Triggs stellen zwei Blockgeometrien vor, rechteckige *R-HOGs* und zirkuläre *C-HOGs*. In R-HOGs werden Zellen in einer rechteckigen Anordnung in einen Block aufgenommen. Die Höhe und Breite eines Blocks in Zellen ist hierbei parametrisierbar. Die typische Anordnung der Zellen in einem Block wird in Abbildung 2.2.2 dargestellt.

C-HOG-Deskriptoren verfügen hingegen über eine runde Form, welche über das *log-polar-Koordinatensystem* beschrieben werden. Geometrisch bestehen C-HOGs aus konzentrisch angeordneten Kreisen mit einer radialen Unterteilung. Die radiale Teilung des innersten Kreises ist nach [DT05] jedoch optional. Die vom C-HOG umschlossene Fläche wird somit in Teilflächen unterteilt, die mehrere Zellen beinhalten können. Die Gradientenhistogramme der Zellen werden innerhalb einer Teilfläche summiert. Anschließend werden die Vektoren der Teilflächen wie in den R-HOGs konkateniert und normiert. In der C-HOG Darstellung gibt es vier wählbare Parameter: Der Radius des Mittelkreises, Anzahl und Radius von äußeren Schichten, und die Anzahl an radialen Unterteilungen.

2.3 K-NEAREST NEIGHBORS

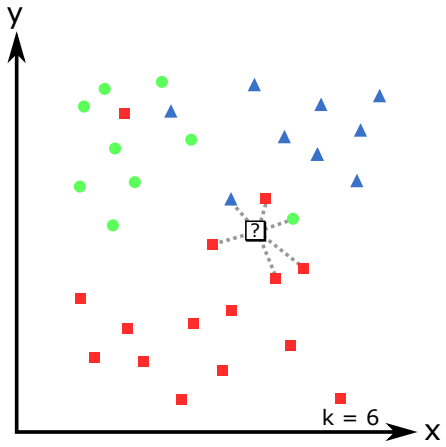


Abbildung 2.3.1: Visualisierung des KNN-Klassifikationsverfahrens, in 2 Dimensionen mit 3 Klassen und $k = 6$

Als erster Klassifikator soll in dieser Arbeit der K-Nearest Neighbor Klassifikator vorgestellt werden. Dieser Klassifikator ordnet ein unbekanntes Sample anhand seiner räumlichen Nachbarschaft einer Klasse zu. Dazu werden die k dem Sample am nächsten liegenden bekannten Datenpunkte ermittelt und ihre jeweiligen Klassenlabels gesammelt. Die Klasse des Samples wird durch einen Mehrheitsentscheid der umliegenden Datenpunkte ermittelt (siehe [TK08], Kapitel 2.6). Der Parameter k muss manuell bestimmt werden, um eine optimale Klassifikationsleistung zu erreichen. Im einfachsten Fall wird k auf 1 gesetzt, wodurch das Sample immer die Klasse seines nächsten Nachbarn erhält.

Die Lernphase des KNN-Klassifikators besteht einzig aus dem Einlesen der Trainingsdaten. Die Klassifikation eines unbekannten Objekts kann allein durch die räumliche Verteilung der Trainingsdaten ermittelt werden. Zur effizienteren Verarbeitung im Computer existieren jedoch Verfahren, welche die Trainingsdaten in eine andere, laufzeit- oder speichereffiziente, interne Repräsentation umwandeln. Nähere Details hierzu finden sich u. a. in [DHS01], Abschnitt 4.5.5.
Zur Ermittlung der Nachbarn muss ein passendes *Distanzmaß* gewählt werden. Dieses Distanzmaß $D(\mathbf{a}, \mathbf{b})$ muss für die Vektoren $\mathbf{a}$, $\mathbf{b}$ und $\mathbf{c}$ folgende Axiome erfüllen (vgl. [DHS01], S.31):

$$\text{Nichtnegativität}: \quad D(\mathbf{a}, \mathbf{b}) \geqslant 0 \tag{2.3.1}$$

$$\text{Reflexivität}: \quad D(\mathbf{a}, \mathbf{b}) = 0 \quad \text{gdw.} \quad \mathbf{a} = \mathbf{b} \tag{2.3.2}$$

$$\text{Symmetrie}: \quad D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a}) \tag{2.3.3}$$

$$\text{Dreiecksungleichung}: \quad D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geqslant D(\mathbf{a}, \mathbf{c}) \tag{2.3.4}$$

Häufig verwendete Distanzmetriken sind die *euklidische Distanz* und der *Cityblock-Abstand*. Details zu diesen und anderen Abstandsmetriken finden sich in [DHS01], Kapitel 4.6. Je nach Verteilung der Daten kann es vorkommen, dass für ein Sample mehr als k Punkte in Betracht kommen, etwa wenn mehrere Nachbarn existieren, die den gleichen Abstand zum Sample haben. Diese Situation kann aufgelöst werden, indem aus den redundanten Datenpunkten zufällig einer ausgesucht wird, welcher für die Klassifikation benutzt wird. Auch ist es möglich, alle redundanten Datenpunkte in die Klassifikation einzubeziehen, und das k für dieses Sample implizit zu erhöhen. Zusätzlich kann die Auswirkung der Nachbarsamples auf die Klassifikation gewichtet werden [JCWJ07], sodass sich verschiedene Nachbarn unterschiedlich

stark auf die Klassifikation auswirken. Ein einfaches Verfahren ist die Gewichtung der umliegenden Punkte durch ihre Abstände zum gesuchten Sample, wodurch zu diesem weiter entfernte Punkte weniger stark ins Gewicht fallen. Dadurch verringert sich die Wahrscheinlichkeit, dass das Sample zu diesen, weiter entfernten Punkten zugeordnet wird.
Unter bestimmten Umständen besteht die Möglichkeit, dass für ein Sample mehrere Klassen bestimmt werden. In diesem Fall muss ein *Tiebreak* vollzogen werden, um die Ambiguität aufzulösen, und das Sample eindeutig einer Klasse zuzuordnen. Ähnlich wie für die Auflösung von redundanten Nachbarn ist ein Ansatz, die Klasse aus allen bestimmten Klassen zufällig auszuwählen (vgl. [Bis06], S.126, Z.5)
Der KNN-Klassifikator ist ein sehr intuitiver Ansatz, der aufgrund seiner Einfachheit in der Praxis häufig Verwendung findet. Ein Nachteil ist der große Speicherbedarf, da potenziell eine große Menge von Merkmalsvektoren gespeichert werden müssen, die je Tausende von Dimensionen enthalten können.

2.4 SUPPORT VECTOR MACHINE

Bei der Support Vector Machine handelt es sich um einen linearen, binären Klassifikator, der zwei Klassen von Samples im n-dimensionalen Raum voneinander trennt. Dies geschieht, indem die SVM eine $n-1$-dimensionale Hyperebene im Raum konstruiert, welche als *decision boundary* ([Bis06] S.179) bezeichnet wird. Wird ein unbekanntes Sample in die SVM zur Klassifikation eingegeben, berechnet diese, auf welcher Seite der Hyperebene das Sample liegt und gibt anhand dieser das Klassenlabel aus. Um mehr als zwei Klassen trennen zu können, existieren mehrere Erweiterungen der SVM, die im Verlauf dieses Abschnitts näher vorgestellt werden. SVMs gehören zu den *linearen Klassifikatoren*, da die Trennebene linear ist (siehe [TK08], Kapitel 3). Im Rahmen der Arbeit wird die SVM hauptsächlich als linearer Klassifikator betrachtet, die SVM lässt sich jedoch auch auf nichtlineare Trennebenen erweitern (siehe [TK08], Kapitel 4.18).

2.4.1 *Lineare Klassifikatoren*

Formell besteht eine n-dimensionale Hyperebene $g(\mathbf{x})$ aus den Punkten im Raum $\mathbf{x}$, für die folgende Gleichung gilt ([TKo8], Kapitel 3.2):

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0 \tag{2.4.1}$$

Der Spaltenvektor $\mathbf{w} = [w_1, w_2, ..., w_n]^T$ beschreibt zusammen mit dem *Bias* w_0 die Ausrichtung der Hyperebene im Raum. $\mathbf{w}$ ist ein Normalenvektor der Ebene, während w_0 den Abstand der Hyperebene zum Ursprung beschreibt. Der Abstand der Hyperebene zum Ursprung ergibt sich aus (vgl. [Biso6], Kapitel 4.1.1):

$$d = -\frac{w_0}{\|\mathbf{w}\|_2} \tag{2.4.2}$$

Der Abstand eines Punktes $\mathbf{x}$ zur Hyperebene lässt sich wie folgt ermitteln:

$$z = \frac{g(\mathbf{x})}{\|\mathbf{w}\|_2} \tag{2.4.3}$$

Die Funktion $g(\mathbf{x})$ gibt also für einen Vektor $\mathbf{x}$ und die durch $\mathbf{w}$ und w_0 beschriebene Hyperebene den normalen Abstand von $\mathbf{x}$ zur Hyperebene aus. Das Vorzeichen des Ergebnisses bestimmt die Entscheidung des Klassifikators: Bei einem negativen Vorzeichen wird die Entscheidung für eine Klasse getroffen, bei einem positiven Vorzeichen wird die andere Klasse gewählt. Für einen beliebigen linearen Klassifikator müssen also, anhand einer Menge von Trainingsvektoren, die Parameter $\mathbf{w}$ und w_0 so gewählt werden, dass die entstehende Ebene die beiden Klassen möglichst genau voneinander trennt.

2.4.2 *Large-margin Klassifikator*

Die Aufgabe der Support Vector Machine ist es, zwischen zwei Klassen eine Trennebene zu ermitteln, und dabei den Abstand aller Datenpunkte einer Klasse zu dieser

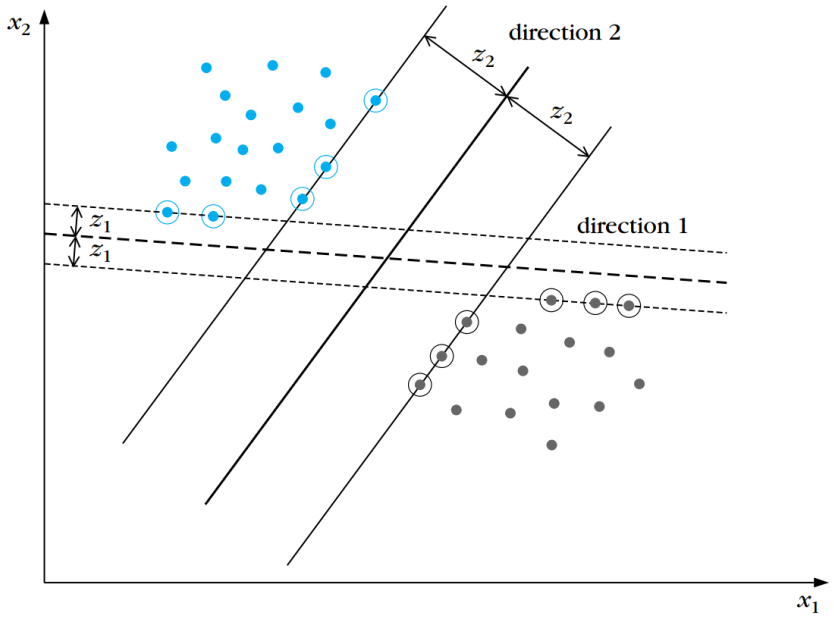


Abbildung 2.4.1: Lineare SVM mit zwei möglichen Trennebenen. direction 2 weist einen deutlich breiteren Rand als direction 1 auf. Die umkreisten Punkte sind die Stützvektoren der angrenzenden Randebenen. Abbildung aus [TK08], S.121.

Hyperebene zu maximieren. Dazu wird um die Hyperebene ein *Rand* (*margin*) berechnet. Dieser Rand besteht aus zwei, zur Trennebene parallelen Hyperebenen, die jeweils im gleichen Abstand auf je einer Seite der Trennebene stehen. Die Randebenen werden analog der Trennebene definiert:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 1 \tag{2.4.4a}$$

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = -1 \tag{2.4.4b}$$

Innerhalb von Gleichung 2.4.4 wird der Abstand zwischen Trennebene und Randebene auf 1 normiert. Die Parameter $\mathbf{w}$ und w_0 müssen entsprechend skaliert werden, um die gewünschte Hyperebene mit einer bestimmten Randbreite zu erzeugen. Der

tatsächliche Abstand zwischen Hyperebene und Randebene ergibt sich aus ([TKo8], S.120):

$$\frac{1}{\|\mathbf{w}\|_2} \tag{2.4.5}$$

Durch den Rand wird eine quantifizierbare Größe geschaffen, die es erlaubt, mehrere Trennebenen gegeneinander zu vergleichen: Aus allen möglichen Trennebenen wird die mit dem breitesten Rand gewählt.

Sind beide Klassen voneinander *linear separierbar*, so kann immer mindestens eine Trennebene bestimmt werden, in deren Rand sich keine Samples befinden. Von allen möglichen Trennebenen verfügt genau eine immer über den breitesten möglichen Rand. Ist der breiteste Rand gefunden existiert für beide Klassen zwangsläufig je ein oder mehr Datenpunkte, die genau auf der jeweiligen Randfläche liegen. Diese Vektoren werden als die *Stützvektoren* bezeichnet, und verschaffen der SVM ihren Namen.
Um die Hyperebene zu berechnen muss ein Optimierungsproblem gelöst werden. Dazu muss zunächst für die Trainingssamples eine passende Darstellung gefunden werden: Jedem Objekt $\mathbf{x}_i$ der Trainingsmenge wird entsprechend seiner Klassenzugehörigkeit ein *Klassenindikator* y_i, bei der SVM entweder -1 oder 1, zugeordnet (vgl. [TKo8], S.121). Aus Gleichung 2.4.5 geht hervor, dass die Breite der Randfläche invers von $\|\mathbf{w}\|_2$ abhängig ist. Es gilt somit, den Abstand der Randebenen so sehr wie möglich zu maximieren, und dabei nicht zu erlauben, dass ein Sample die Randfläche überschreitet. Das entstehende Optimierungsproblem setzt sich wie folgt zusammen (vgl. [TKo8], S.121):

$$\begin{aligned} \underset{\mathbf{w}, w_0}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i \times (\mathbf{w}^T \mathbf{x}_i + w_0) \geqslant 1 \end{aligned} \tag{2.4.6}$$

Effiziente Darstellungsformen und Lösungsverfahren für das Optimierungsproblem sind der Fachliteratur ([TKo8], S.121ff, [Mur12], S.328ff) zu entnehmen. Eine so erstellte SVM lässt es nicht zu, dass ein Trainingssample in den Rand fällt, oder die Trennebene überschreitet. Eine solche SVM bezeichnet man als *hard-margin* SVM. In vielen Fällen ist die Trainingsmenge jedoch nicht praktisch linear trennbar, wodurch

sich durch das oben beschriebene Optimierungsproblem keine Lösung ermitteln lässt. Deshalb müssen die Randbedingungen zum Teil gelockert werden. Zu diesem Zweck wird im Folgenden die *soft-margin* SVM beschrieben.

2.4.3 *Soft-margin SVM*

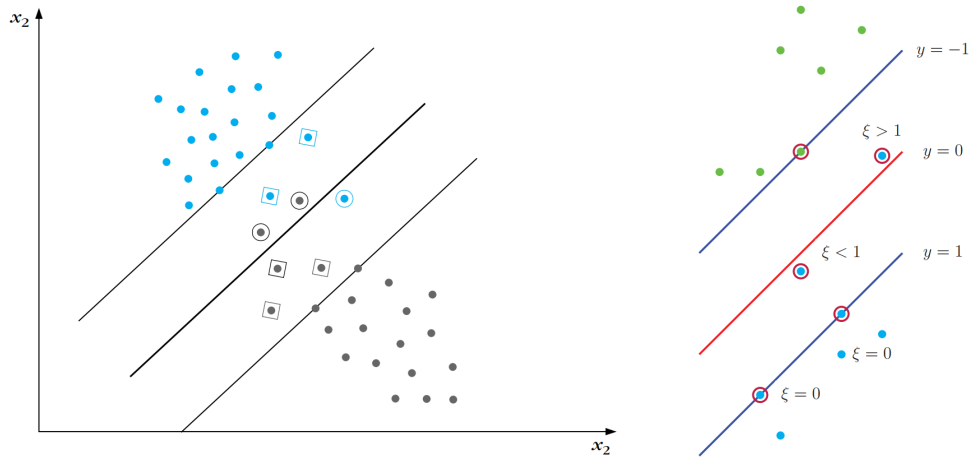
Die soft-margin SVM bedient sich dem Konzept der Schlupfvariablen (*slack variables*) um zu erlauben, dass die in Gleichung 2.4.6 eingeführten Randbedingungen teilweise verletzt werden können. Für jedes Sample $\mathbf{x}_i$ wird eine zugehörige Slackvariable ξ_i eingeführt. Die Slackvariablen sind wie folgt definiert (vgl. [Mur12], S.501f). Der Klassenindikator y_i und die Abstandsfunktion $g(\mathbf{x})$ sind wie zuvor definiert.

$$\xi_i = \begin{cases} 0, & \text{if } y_i \times g(\mathbf{x}_i) \geqslant 1 \\ |y_i - g(\mathbf{x}_i)|, & \text{otherwise} \end{cases} \tag{2.4.7}$$

Ist ξ_i also gleich 0, so ist der Punkt richtig klassifiziert. Zwischen 0 und 1 liegt der Punkt auf der richtigen Seite der Hyperebene, aber innerhalb des Randes. Für $\xi_i > 1$ ist der Punkt eindeutig falsch klassifiziert. Das überarbeitete Optimierungsproblem setzt sich wie folgt zusammen ([Mur12], S.501):

$$\begin{aligned} \underset{\mathbf{w}, w_0, \xi_i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_i \xi_i \\ \text{subject to} \quad & \xi_i \geqslant 0,\ y_i \times (\mathbf{w}^T\mathbf{x}_i + w_0) \geqslant 1 - \xi_i \end{aligned} \tag{2.4.8}$$

In Gleichung 2.4.8 wird gezeigt, wie die Randbedingung gelockert wird. Verletzt ein Sample $\mathbf{x}_i$ die Randbedingung, so kann es stets durch ein passendes ξ_i ausgeglichen werden. Dennoch gilt es die Gesamtsumme der ξ_i zu minimieren. Zusätzlich werden die ξ_i mit einem Regulierungsparameter C versehen, der zusätzlich den Einfluss der Slackvariablen regelt. Ein hoher C-Parameter gewichtet alle ξ_i enorm, wodurch das Optimierungsproblem tendenziell schmale Ränder bevorzugt. Ein niedriges C erlaubt hingegen sehr große ξ_i, wodurch breitere Ränder entstehen. Der C-Parameter ist frei wählbar und muss von Hand bestimmt werden.



(a) (b)

Abbildung 2.4.2: (a): In der abgebildeten SVM sind beide Klassen nicht linear separierbar. Die mit einem Quadrat umrandeten Punkte befinden sich in der Randfläche, die umkreisten Punkte überschreiten den Rand. Abbildung aus [TK08], S.124. (b): Darstellung von Slackvariablen in einer SVM für die untere Hälfte des Bildes. Abbildung aus [Mur12], S.500

2.4.4 *Mehrklassen-SVMs*

Die bisher vorgestellten hard-margin und soft-margin SVMs sind nur in der Lage, zwei Klassen zu unterscheiden. Um zehn Klassen von Hausnummern voneinander abzugrenzen werden hier zwei mögliche Ansätze vorgestellt, die SVM auf mehrere Klassen zu erweitern:

- 1 gegen 1
- 1 gegen n

Im Fall 1 gegen 1 werden alle Klassen paarweise miteinander verglichen. Dazu werden für m verschiedene Klassen $m(m-1)/2$ Trennebenen erzeugt. Ein einzuordnendes Sample wird mit allen Trennebenen verglichen, und der Klasse zugeordnet, die von allen Klassifikatoren am häufigsten errechnet wird. Dieser Ansatz ist jedoch sehr rechenintensiv, da die Anzahl der zu berechnenden Hyperebenen quadratisch mit m wächst. Zudem ist es wie auch beim KNN-Klassifikator möglich, dass einem Sample mehrere Klassen mit der gleichen Häufigkeit zugewiesen werden (vgl. [Bis06],

S.183). Auch hier ist es notwendig, die einen Tiebreak durchzuführen.

Es ist auch möglich, jeweils immer alle Elemente einer Klasse von allen Nichtelementen zu trennen. So ist die Funktionsweise des 1 gegen n Klassifikators. Auch hier besteht für ein Sample die Gefahr der Doppelklassifikation, weshalb eine Gewichtung der Klassenzugehörigkeit sinnvoll ist. Für eine Klasse K_i wird somit die folgende Trennebene ermittelt (vgl. [Bis06], S.183):

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = 0 \tag{2.4.9}$$

Ein unbekanntes Sample $\mathbf{x}_j$ wird der Klasse K_i zugewiesen, für die sein Abstand zu der Trennebene der Klasse am größten ist. Dafür wird für jede der Klassen der Wert der Abstandsfunktion $g_i(\mathbf{x}_j)$ berechnet, und das Maximum ausgesucht.

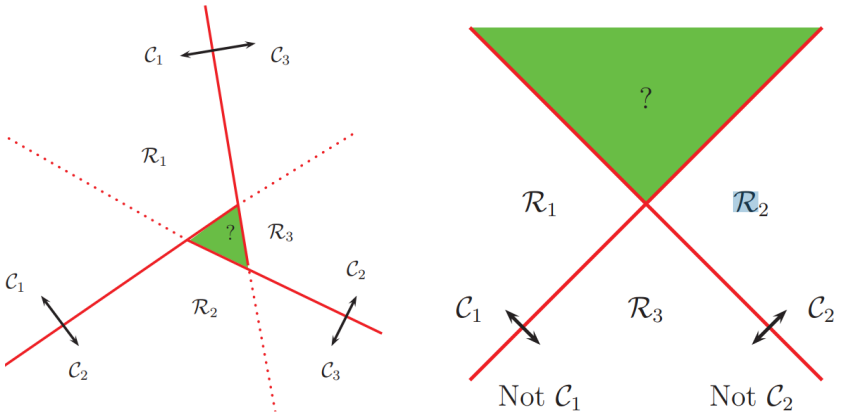


Abbildung 2.4.3: (a): Paarweise lineare Trennung von Klassen (1 gegen 1). Mittig entsteht ein Bereich, der keiner Klasse eindeutig zugeordnet werden kann.
(b): Trennung von zwei Klassen im 1 gegen n-Verfahren. Oben entsteht eine Fläche, die zu keiner Klasse gehört. Abbildungen entnommen aus [Mur12], S.500.

2.4.5 *Nichtlineare SVMs*

Zuletzt soll in diesem Abschnitt kurz auf nichtlineare SVMs eingegangen werden. In bestimmten Fällen, z. B. das *XOR-Problem* ([DHS01], Abschnitt 5.11.1, Beispiel 2), ist

es nicht möglich, einen linearen Klassifikator zur Trennung heranzuziehen. Eine Lösung für das Problem ist es, die Trainingsmenge in einen höherdimensionalen Raum abzubilden, und in diesem einen linearen Klassifikator zu berechnen. Durch den Einsatz von *Kernelfunktionen* (siehe [Bis06], Kapitel 6) lassen sich auch nichtlineare SVMs effizient berechnen. Näheres hierzu findet sich u. a. in [TK08], Kapitel 4.
Für ein bestimmtes Problem muss ein passende Kernelfunktion bestimmt werden. Verschiedene gängige Kernelfunktionen werden in [Mur12], Kapitel 14.2 erläutert.

3 VERWANDTE ARBEITEN

Das Thema der Szenentexterkennung hat viele praktische Anwendungsbereiche (siehe Kapitel 1), weshalb es in der Forschung bereits häufig untersucht wurde. Um aus natürlichen Szenenbildern Textelemente auszulesen ist es zunächst notwenig, in den Bildern Textregionen zu lokalisieren, und in den Regionen den Textinhalt zu erkennen (vgl. [YD15]). Bei einigen Verfahren findet zunächst eine Segmentierung der Textregionen in Zeichen oder Wörter statt, während andere Verfahren den gesamten Text eines Textbereiches in einem Schritt erkennen. Um diese Aufgaben zu bewältigen wurden viele verschiedenartige Verfahren, Merkmalsrepräsentationen und Datensätze vorgestellt [ZYB16]. In diesem Kapitel sollen thematisch und methodisch verwandte Arbeiten vorgestellt und diskutiert werden. Insbesondere werden im Folgenden existierende Verfahren zur Hausnummernerkennung vorgestellt.
Ein weiteres, in der Objekt- und Szenentexterkennung wichtiges Thema ist Augmentation. Augmentierte und synthetische Datensätze werden eingesetzt, um die Menge der Trainingsobjekte künstlich zu vergrößern. Dies ist nützlich in Problemen, in denen nur wenige natürliche Trainingsbeispiele vorhanden sind, oder einzelne Klassen besonders selten vorkommen. Zusätzlich benötigen künstliche Bilder keine manuelle Annotation, wodurch der Aufwand in der Erstellung eines Datensatzes enorm gesenkt werden kann. In Abschnitt 3.2 werden Arbeiten vorgestellt, in denen für eine Klassifikationsaufgabe eine augmentierte Trainingsmenge herangezogen wurde. Es wird kurz auf die Methodik, die angewandten Augmentationen, und die Ergebnisse eingegangen.

3.1 HAUSNUMMERNERKENNUNG

Das Erkennen von Hausnummern in Szenenbildern lässt sich in den größeren Bereich der Szenentexterkennung einordnen. Die Erkennung von Hausnummern zeigt ebenfalls Parallelen zur Erkennung von handschriftlichen Zahlen auf, da in beiden Fällen die gleichen Objekte, die indisch-arabischen Ziffern von 0 bis 9, klassifiziert werden. Anders als handschriftliche Zahlen ist die Erkennung von Hausnummern jedoch mit den für Szenentext typischen Herausforderungen versehen, weshalb für handschriftliche Zahlen erfolgreiche Verfahren nicht zwangsläufig auf Hausnummern übertra-

gen werden können. In [NWC+11] veröffentlichen Netzer, et. al eine wichtige Arbeit im Bereich der Hausnummernerkennung. Die Autoren stellen den *Street View House Numbers* (SVHN) Datensatz vor, welcher eine Vielzahl von fotografisch erfassten Hausnummern aus Google Street View enthält.

In der Arbeit befassen sich Netzer et al. mit der Detektion von Hausnummern in Fotos und der anschließenden Klassifikation der Hausnummern. Zur Detektion nutzen Netzer et al. ein *gleitendes Fenster*, welches dem Ansatz [FCA+09] nachempfunden ist. Netzer et al. erkennen die Hausnummern auf Ziffernebene, weshalb die detektierten Hausnummernbereiche zusätzlich in einzelne Ziffern *segmentiert* werden müssen. Details zu diesem Vorgang sind der Arbeit von Netzer et al. zu entnehmen. Die Erkennung wird auf 32×32 Pixel großen Ziffernbildern durchgeführt. Netzer et al. evaluieren für die Ziffernbilder verschiedene Merkmalsrepräsentationen, darunter die HOG-Deskriptoren [DT05], Binary Features [KWTM97] und zwei *feature learning* Algorithmen; *stacked sparse auto-encoders* und ein auf *K-Means* basierender Algorithmus [NWC+11]. Die extrahierten Merkmale werden durch eine lineare SVM klassifiziert. Einzelheiten zu der Klassifikationsphase finden sich in der Arbeit von Netzer et al [NWC+11]. Die Ergebnisse der Arbeit sind in Tabelle 3.1.1 aufgelistet.

Die Arbeit von Netzer et al. ist im Bereich der Hausnummernerkennung durch die Etablierung des SVHN-Datensatzes von großer Bedeutung. Der SVHN-Datensatz wurde bereits in mehreren anderen Arbeiten eingesetzt, u. a. in [BMK14], [GBI+13] und [SCL12]. Auch in dieser Bachelorarbeit wird der SVHN-Datensatz zur Evaluation verwendet. Weiterhin nutzen Netzer et al. die HOG-Deskriptoren und eine lineare SVM, welche ebenfalls Gegenstand dieser Bachelorarbeit sind. In Abschnitt 4.1 dieser Arbeit wird genauer auf die Zusammensetzung und die Details des SVHN-Datensatzes eingegangen.

Abbildung 3.1.1: Hausnummernbilder aus dem SVHN-Datensatz, vorgestellt in [NWC+11].

HOG	85,0%
Binary Features (WDCH)	63,3%
K-Means	90,6%
Stacked Sparse Auto-Encoders	89,7%
Menschliche Erkennungsleistung	98,0%

Tabelle 3.1.1: Klassifikationsergebnisse der Experimente aus [NWC+11] für die Hausnummernerkennung.

3.2 AUGMENTATION

Die Idee liegt nahe, in visuellen Mustererkennungsaufgaben synthetisch Trainingsbeispiele zu erzeugen. Ist es möglich, künstlich ein Bild zu erzeugen, das für den menschlichen Betrachter nicht von einem echten Bild zu unterscheiden ist, so ist es auch denkbar, mit diesem Bild einen maschinellen Klassifikator zu trainieren. Es können zwei Methoden angewendet werden, um eine datenaugmentierte Trainingsmenge zu erzeugen: Ein bestehender Datensatz kann erweitert werden, indem einzelne Elemente dupliziert und augmentiert werden, oder Trainingsbeispiele werden von Grund auf synthetisch erzeugt.
Der Einsatz von Datenaugmentationen bietet viele potenzielle Vorteile: Die größere Anzahl an Trainingsbildern kann zu besseren Klassifikationsergebnissen führen (siehe [Bai93], [WGSM16]), und die Generalisierung des Klassifikators verbessern [CMGS10]. Es haben sich bereits viele Arbeiten mit augmentierten oder synthetischen Daten befasst. Augmentationen können verschiedene Ziele verfolgen, zum Beispiel das Erweitern der Trainingsmenge, oder das gezielte Einfügen bestimmter Fehler in ein Bild, um den Klassifikator auf das Vorkommen solcher Fehler vorzubereiten.
Baird stellt in [Bai93] einen Datensatz aus gedruckten Buchstaben vor, der mit augmentierten Daten versehen ist. Die augmentierten Trainingsbilder sind zu verschie-

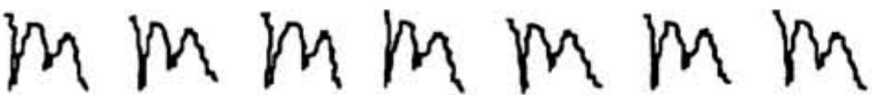
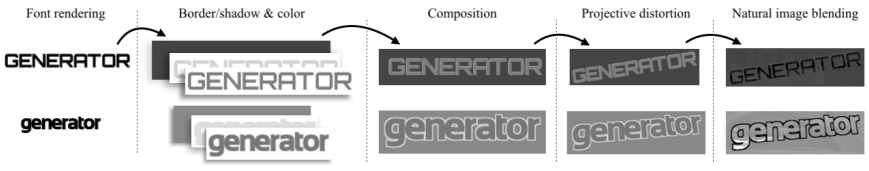


Abbildung 3.2.1: Verschiedene synthetisch erzeugte Varianten des selben Buchstabens. Abbildung aus [YLW97]

denen Graden verfälscht und unkenntlich gemacht worden, wodurch Druckfehler in Buchstaben simuliert werden.
Ein häufig eingesetztes Augmentationsverfahren für Buchstaben und Zahlen ist das zufällige elastische Transformieren der Striche, welches auch *warping* genannt wird. Yaeger et al. trainieren in [YLW97] ein künstliches neuronales Netz (ANN) mit zufällig transformierten Zeichen. Ein Beispiel dafür ist in Abbildung 3.2.1 zu sehen.

Auch zur Szenentexterkennung wurden bereits häufig augmentierte oder synthetische Daten herangezogen. De Campos et al veröffentlichten in [CBV09] den *chars74k*-Datensatz, der zu Teilen aus synthetisch generierten Daten besteht. Dieser Datensatz wird zur Erkennung von Buchstaben verwendet, und enthält fotografisch erfasste, handschriftlich geschriebene und digital generierte Zeichenbilder.
In Jaderberg et al. [JSVZ14] werden rein synthetische Textbilder generiert, mit denen ein neuronales Netz (CNN) trainiert wird: Es wird mit randomisierten Parametern ein Textelement auf ein leeres Bild gezeichnet. Schriftart, Farbe und Schriftparameter werden zufällig ausgewählt, und es kann ein Schatten unter den Text gezeichnet werden. Die Textelemente werden perspektivisch verzerrt, und auf ein natürliches Szenenbild gelegt. Anschließend kann das Bild zusätzlich durch Rauschen, Unschärfe und JPEG-Kompressionsartefakte weiter verfälscht werden. Mit den synthetisch generierten Wortbildern wird das neuronale Netz trainiert. Die Leistung des neuronalen Netzes prüfen Jaderberg et al. mit mehreren natürlichen Datensätzen.
In [JVZ14] werden zur Zeichenerkennung in Bildern die Trainingsdaten durch Rotationen und das Hinzufügen von Rauschen leicht augmentiert. In [ZWWW15] werden



(a)

(b)

Abbildung 3.2.2: (a): Synthetischer Wortbildgenerierungsprozess nach [JSVZ14] (b): Zufällige synthetisch generierte Wortexemplare. Abbildungen aus [JSVZ14].

hingegen für Augmentation zufällige Skalierungen und Rotationen eingesetzt, die auf die 32×32 Pixel großen Trainingsbilder angewandt werden. Aus den Bildern werden darauf zufällig 28×28 Pixel große Bereiche ausgeschnitten.
Auch in der Hausnummernerkennung wurden teilweise bereits Augmentationsverfahren durchgeführt: Goodfellow et al. [GBI+13] klassifizieren in ihrer Arbeit Hausnummernbilder als Sequenz von Zahlen. Die Autoren erzeugen für jedes Trainingsbild mehrere augmentierte Varianten, indem ein Bild zufällig um einige Pixel verschoben wird. Das gleiche Augmentationsverfahren wird ebenfalls in Ba et al. [BMK14] eingesetzt.

4 METHODIK

In diesem Kapitel wird der Versuchsaufbau der Arbeit vorgestellt. Das Ziel ist, eine möglichst hohe Klassifikationsleistung in der Erkennung von Hausnummern zu erreichen. Es gilt also, einen Klassifikator mit einer Trainingsmenge von Hausnummernbildern zu trainieren. Um die Leistung des Klassifikators zu evaluieren, muss dieser mit einer weiteren Menge von Bildern getestet werden. In dieser Arbeit wird dafür der SVHN-Datensatz [NWC+11] verwendet, welcher aus getrennten Trainings- und Testmengen besteht. Details zum SVHN-Datensatz werden in Abschnitt 4.1 erläutert. Aus den SVHN-Bildern werden auch augmentierte Bilder erstellt, die in verschiedenem Maßen der Trainingsmenge zugeführt werden. Alle Bilder müssen in einem bestimmten, einheitlichen Format vorliegen und ggf. vorverarbeitet werden. Darauf werden aus den Bildern die jeweiligen HOG-Deskriptoren ermittelt. Wie in Abschnitt 2.2 angemerkt, ist die Beschaffenheit der HOG-Deskriptoren von einer Anzahl von Hyperparametern abhängig, die einen großen Einfluss auf die Klassifikationsleistung haben. Im Rahmen dieser Arbeit werden verschiedene Kombinationen von Hyperparametern verwendet und evaluiert. Mit den HOG-Merkmalsvektoren gilt es dann, einen Klassifikator zu trainieren. Es werden der KNN-Klassifikator und eine lineare Support Vector Machine untersucht, welche miteinander verglichen werden. Beide Klassifikatoren sind ebenfalls von einer Menge von Hyperparametern abhängig, welche die Erkennungsleistung beeinflussen. Auch hier gilt es, mehrere Konfigurationen der Klassifikatoren zu evaluieren.
Um die Leistung der Klassifikatoren zu prüfen wird zunächst eine Kreuzvalidierung über die Trainingsmenge durchgeführt. Die Klassifikatoren, die in der Kreuzvalidierung am besten abschneiden, können dann auf dem Test-Datensatz evaluiert werden. Auf die Einzelheiten der Versuchsphase wird in Abschnitt 4.2 eingegangen. Die Ergebnisse der Kreuzvalidierung und der Testphase werden in Kapitel 5 aufgeführt.

4.1 SVHN-DATENSATZ

Der SVHN-Datensatz wurde 2011 in [NWC+11] vorgestellt. Der Datensatz besteht aus einer Menge von Bildern von Hausnummern, die aus Aufnahmen von Google Street View gesammelt wurden. In den Fotos wurden automatisch die Zahlenregio-

nen lokalisiert. Hierzu wurde ein Zahlendetektor nach [FCA+09] verwendet, der auf einem gleitenden Fenster basiert. Der Detektor wurde so konfiguriert, dass in den Bildern viele potenzielle Zahlenregionen erkannt werden, auch wenn dabei viele Falscherkennungen (*false positives*) entstehen. Aus den potenziellen Hausnummernregionen wurden darauf manuell die erkennbaren Zahlen ermittelt, welche den Bildern als *ground truth* beiliegen. Falscherkennungen wurden dabei manuell aus der Menge entfernt.

Der SVHN-Datensatz wird in zwei Formaten bereitgestellt, in ganzen Zahlen und als einzelne Ziffern. Die ganzzahligen Bilder enthalten ganze Hausnummern, welche ggf. mehrere Ziffern enthalten. Die Bilder sind mit ihren enthaltenen Ziffern sowie mit den Koordinaten und Ausmaßen (*bounding boxes*) der einzelnen Ziffern annotiert. Ebenfalls liegen die Ziffern einzeln in getrennten Bildern vor, welche jeweils 32×32 Pixel groß sind. Die Ziffern wurden hierbei skaliert, damit sie in das Bildfenster hineinpassen. Nicht immer sind auf den 32×32 Pixel großen Bildern die einzelnen Ziffern sauber voneinander getrennt: Es kommt häufig vor, dass links oder rechts von der gesuchten Ziffer die Ränder angrenzender Ziffern zu sehen sind. Das kleine ziffernbasierte Format ist dem des *MNIST-Datensatzes* nachempfunden, welcher handschriftliche Ziffern enthält [NWC+11]. Die Bilder liegen im RGB-Farbraum vor.

Der SVHN-Datensatz ist in drei Teildatensätze unterteilt. In Tabelle 4.1.1 werden die Größen der Datensätze als Anzahl enthaltener Ziffern gezeigt. Es ist zu beachten, dass, falls ganze Hausnummern betrachtet werden, mehrere Ziffern in einem Bild enthalten sein können.

SVHN TRAIN	73 257
SVHN TEST	26 032
SVHN EXTRA	531 131

Tabelle 4.1.1: Die drei Teildatensätze des SVHN-Datensatzes und die jeweilige Anzahl enthaltener Ziffern. Datensatz aus [NWC+11]

SVHN TRAIN wird von Netzer et al. als Trainingsmenge verwendet. SVHN EXTRA enthält im Vergleich zu SVHN TRAIN deutlichere Bilder, die einfacher zu Klassifizieren sind [NWC+11]. SVHN EXTRA kann zusätzlich zum Trainieren herangezogen werden. SVHN TEST wird von Netzer et al. als Testmenge verwendet.
Die Klassen sind innerhalb der Datensätze nicht gleichmäßig verteilt. In Abbildung 4.1.1

zeigt sich, dass die Klassen 1 und 2 verhältnismäßig überrepräsentiert sind, während die größeren Ziffern und die 0 seltener vorkommen.

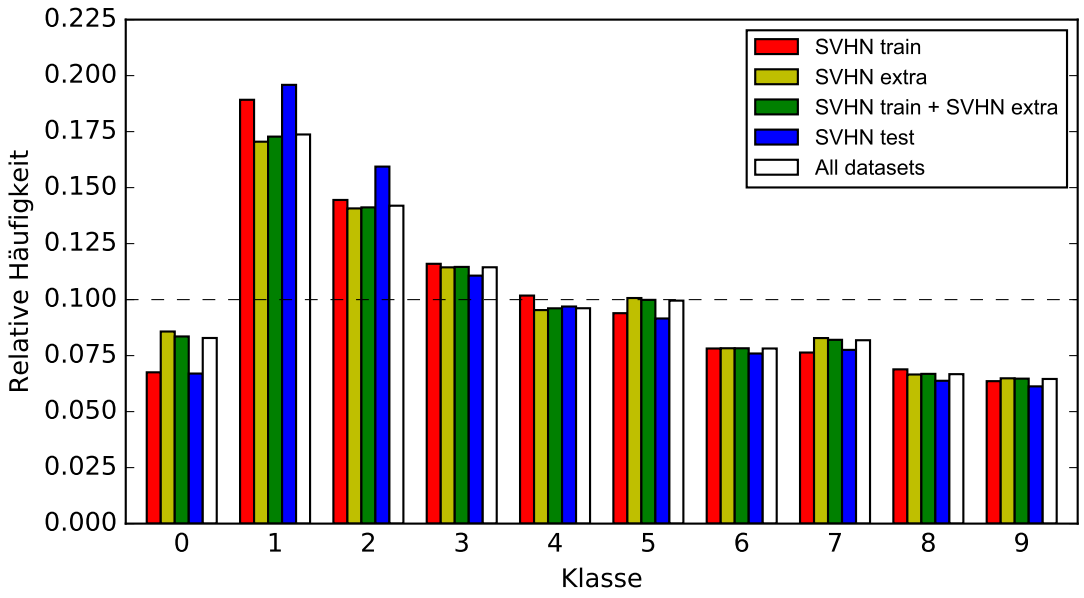


Abbildung 4.1.1: Die relative Klassenverteilung der Teile des SVN-Datensatzes.

4.2 VORGEHEN

Der Versuchsaufbau gliedert sich in zwei Phasen: In der ersten Phase werden der KNN-Klassifikator und die Support Vector Machine für die Hausnummernerkennung verglichen, und die optimalen Hyperparameter für diese gefunden. Der Klassifikator wird auf SVHN TRAIN trainiert und darauf in einer Kreuzvalidierung evaluiert. In der zweiten Phase wird die Trainingsmenge um weitere Bilder ergänzt, wozu SVHN EXTRA und zusätzlich augmentierte Bilder verwendet werden. Das Ziel ist es, den manuellen Annotationsaufwand für die Trainingsbilder zu verringern, und dabei eine möglichst hohe Klassifikationsleistung zu behalten. Zum Klassifizieren werden der beste Klassifikator und die optimale Merkmalsrepräsentation aus Phase 1 herangezogen. Die Einzelheiten werden im Folgenden besprochen.

4.2.1 *Ermitteln des Klassifikators*

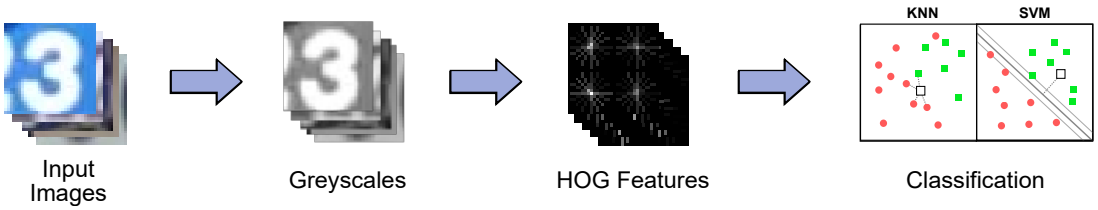


Abbildung 4.2.1: Darstellung des Versuchsaufbaus. Aus den Trainingsbildern werden zunächst die Graustufenbilder berechnet, und aus diesen die HOG-Deskriptoren. Darauf wird mit den Bildern ein Klassifikator trainiert.

Die hier geführten Experimente richten sich nach der Arbeit von Netzer et al. ([NWC+11]). Als Trainingsmenge wird SVHN TRAIN genutzt. Wie in [NWC+11] werden hierzu die 32×32 Pixel großen Ziffernbilder verwendet, welche zuvor in Graustufenbilder umgewandelt werden. Der Helligkeitswert eines Bildpixels lässt sich über die folgende Formel berechnen. Für die Farbwerte R, G und B an einer Stelle im Bild ist die entsprechende Luminanz Y:

$$Y = 0.2125\,R + 0.7154\,G + 0.0721\,B \tag{4.2.1}$$

Die Koeffizienten für R, G und B wurden in [BT90] für den allgemeinen Gebrauch empfohlen. Es ist auch möglich, andere Koeffizienten zu wählen, welche ggf. zu leicht unterschiedlichen Ergebnissen für die Grauwertbilder führen. Näheres zu den theoretischen Hintergründen der Farbmodelle lässt sich aus [NFHS12], Kapitel 3 und [Pri15], Kapitel 3.2 entnehmen.

Aus den vorverarbeiteten Bildern werden nun nach dem Verfahren aus [DT05] die HOG-Deskriptoren berechnet. Auf dem Bild kann zunächst, wie in [DT05] vorgeschlagen, eine Gammakorrektur vorgenommen werden. Dazu kann ein Bild auf die Quadratwurzel seiner Pixelwerte abgebildet werden: $x \to \sqrt{x}$. Aus dem Bild wird dann, wie zuvor beschrieben, das Gradientenbild extrahiert. Im Rahmen der Experimente werden die quadratischen R-HOG-Deskriptoren verwendet, welche auf rechteckig geformten Zellen berechnet werden. Die Größe der Zellen in $x \times y$ Pixeln ist hierbei parametrisierbar, die Anzahl an Werten im Gradientenhistogramm ebenso. Das

Abbildung 4.2.2: 24 zufällig aus SVHN TRAIN entnommene Ziffern.

Histogramm wird hierbei als nicht vorzeichenbehaftet gewertet. Darauf werden die Zellen wie beschrieben zu HOG-Deskriptoren zusammengefasst und normiert, wozu die in Abschnitt 2.2 beschriebenen Normierungsverfahren verwendet werden. Auch die Anzahl an Zellen $i \times j$ innerhalb eines Blocks in beide Richtungen ist parametrisierbar. Sind i oder j hierbei größer als 1 liegt eine Überschneidung der Blöcke vor. Die entstandenen Blöcke werden darauf zu einem Merkmalsvektor konkateniert. Die Dimensionalität des Merkmalsvektors hängt von der Auflösung des Bildes, der Größe der Zellen, der Größe der Blöcke, und der Anzahl an Werten in jedem Zellenhistogramm ab.

Nachdem alle Bilder verarbeitet wurden liegt eine Menge von HOG-Merkmalsvektoren vor, mit denen ein Klassifikator trainiert werden kann. Als Klassifikator wird entweder eine lineare Support Vector Machine oder ein KNN-Klassifikator erzeugt. Für die Support Vector Machine lässt sich der Parameter C (siehe Unterabschnitt 2.4.3) einstellen. Da insgesamt 10 Klassen betrachtet werden, muss für die SVM eine geeignete Mehrklassenstrategie gewählt werden. Dazu wird entweder das 1 gegen 1 oder das 1 gegen n Verfahren (siehe Unterabschnitt 2.4.4) eingesetzt.
Für den KNN-Klassifikator ist der Parameter k von großer Bedeutung für das Klassifikationsergebnis. Als Distanzmaße werden der L2-Abstand (euklidische Distanz) und der L1-Abstand (Cityblock-Abstand) genutzt, welche beides Formen der *Minkowski*-Metrik sind ([DHS01], S.31). Die Abstände für n-dimensionale Vektoren definieren sich wie folgt:

- Cityblock-Abstand

$$L_1(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i| \tag{4.2.2a}$$

- Euklidische Distanz

$$L_2(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{n} |a_i - b_i|^2 \right)^{1/2} \tag{4.2.2b}$$

Zudem werden zwei verschiedene Gewichtungsverfahren verwendet: Die Punkte werden entweder einheitlich gewichtet, sodass die Stimme aller k Nachbarn den gleichen Einfluss auf die Klassifikation hat. Alternativ wird der Einfluss der Nachbarpunkte $\mathbf{X}_i$ auf den inversen Abstand zum Sample $\mathbf{z}$ abgebildet, sodass die die Stimme des Punktes $\mathbf{X}_i$ folgenden Wert hat: $1/L(\mathbf{X}_i, \mathbf{z})$.

Der Klassifikator wird nun in einer fünffachen Kreuzvalidierung evaluiert. Die Trainingsmenge SVHN TRAIN wird nach der Merkmalsberechnung in 5 gleichgroße Teilmengen partitioniert. Der Klassifikator wird in 5 Durchläufen mit je 4 der 5 Teilmengen trainiert und darauf auf der verbleibenden Teilmenge geprüft (vgl. [Bis06], S.32f). Für jedes Element der Testmenge wird vom Klassifikator eine Vorhersage der Klasse getroffen, die mit dem tatsächlichen bekannten Klassenlabel verglichen wird. Das Verhältnis der richtig klassifizierten Testobjekte zu der Gesamtgröße der Testmenge ergibt das Evaluationsergebnis für den Fold. Aus den Ergebnissen aller 5 Folds kann darauf das arithmetische Mittel gebildet werden. Somit werden bei jedem Durchlauf stets andere Elemente als Testelemente verwendet.

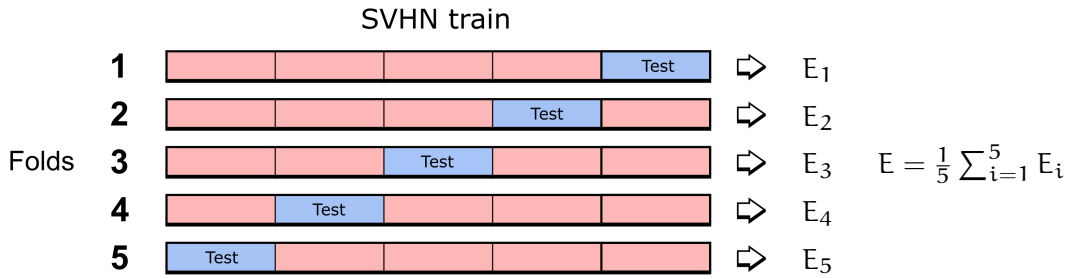


Abbildung 4.2.3: Grafische Darstellung der fünffachen Kreuzvalidierung. Das als Test markierte Segment wird stets als Testmenge eingesetzt. Das Ergebnis E der Kreuzvalidierung ist das arithmetische Mittel der Ergebnisse aller Folds E_i. Abbildung nach [Ras15], S.176.

Das Ergebnis der Kreuzvalidierung gibt einen Einblick in die Leistung des Klassifikators. Durch die Kreuzvalidierung können die besten Parameterkombinationen

ermittelt werden. Die am besten abschneidenden Parameterkonfigurationen werden darauf auf der Menge SVHN TEST final evaluiert. Genauere Details zu den Evaluationsergebnissen, sowie die besten Parameterkombinationen für SVM und KNN werden in Abschnitt 5.2 besprochen.

4.2.2 *Erweiterung der Trainingsmenge*

Nachdem in Unterabschnitt 4.2.1 mittels der Kreuzvalidierung eine optimale Parameterkombination für den Klassifikator ermittelt wurde, gilt es jetzt, die Trainingsmenge zu erweitern. Dazu wird zusätzlich der SVHN EXTRA Datensatz herangezogen, und es werden Bilder aus SVHN TRAIN und SVHN EXTRA ausgewählt und augmentiert. Es gilt, effektive Augmentationsverfahren zu finden, und ein angemessenes Verhältnis zwischen natürlichen und augmentierten Bildern zu bestimmen. In Abschnitt 4.3 werden die für die Arbeit relevanten Augmentationsverfahren aufgeführt und beschrieben.

Zur Klassifikation wird folglich eine Trainingsmenge aus SVHN TRAIN, SVHN EXTRA, und augmentierten Bildern konstruiert. Die gesamte Trainingsmenge enthält bis zu 300 000 Bilder, die zu einem gewissen Verhältnis durch Augmentation entstehen. Der natürliche Anteil der Bilder wird immer zunächst aus den Bildern von SVHN TRAIN entnommen. Reicht dieser Datensatz nicht aus, um die gewünschte Anzahl an natürlichen Bildern zu erhalten, werden zusätzliche Bilder aus SVHN EXTRA ausgewählt und der Trainingsmenge beigefügt. Darauf wird aus der entstandenen Trainingsmenge die gewünschte Menge von augmentierten Bildern berechnet. Für eine Trainingsmenge, die aus n Bildern besteht, werden folgende Zusammensetzungen betrachtet:

- **Keine Augmentation**: Alle n Bilder werden aus SVHN TRAIN und ggf. SVHN EXTRA entnommen.
- **Verdoppelung der Trainingsmenge**: Es werden $n/2$ natürliche Bilder genommen. Für jedes Bild wird je eine Augmentation erstellt und der Menge hinzugefügt.
- **Vervierfachung der Trainingsmenge**: Es werden $n/4$ natürliche Bilder genommen. Für jedes Bild werden je drei Augmentationen erstellt und der Menge hinzugefügt.

Für alle diese Zusammensetzungen wurden unterschiedlich große Trainingsmengen konstruiert. Die größte betrachtete Trainingsmenge verfügt insgesamt über 300 000 Elemente. Die Menge besteht entweder ganz, zur Hälfte oder zu einem Viertel aus natürlichen Bildern.

Mit der erweiterten Trainingsmenge wird nun der zuvor bestimmte Klassifikator trainiert. Bei diesem Schritt wird auf eine Kreuzvalidierung verzichtet; stattdessen wird direkt eine Evaluation auf SVHN TEST durchgeführt. Die optimalen Augmentierungsverfahren sowie das beste Verhältnis von augmentierten zu natürlichen Bildern wird in Abschnitt 5.4 aufgeführt.

4.3 AUGMENTATIONSVERFAHREN

Zur Augmentation der Bilder wird eine Reihe von grafischen Augmentationsverfahren verwendet, die das Bild in einer gewissen Weise verändern. Die Verfahren lassen sich vollautomatisch und randomisiert auf eine große Menge von Bildern anwenden, wodurch sich in kurzer Zeit eine Vielzahl von neuen Bildern erzeugen lässt. Es muss Acht gegeben werden, ein Bild nicht so sehr zu verändern, dass es unkenntlich wird: Sind die Trainingsbilder zu stark verfälscht, kann dies die Klassifikationsleistung beeinträchtigen. Im Folgenden werden die für die Arbeit relevanten Augmentationsverfahren vorgestellt. Ähnlich wie auch die Klassifikationsverfahren sind viele der Augmentationen parametrisierbar, und werden mit unterschiedlicher Stärke auf ein Bild angewendet. Für jedes Bild werden seine zugehörigen Augmentationsverfahren und die zugehörigen Parameter zufällig aus einem festgelegten Wertebereich ausgewählt. Ein Augmentationsverfahren kann formell als Abbildung betrachtet werden, die einer Bildmatrix einer gewissen Größe eine neue Bildmatrix zuordnet. Die verschiedenen Augmentationsverfahren werden sequentiell auf einem Bild durchgeführt. Hierbei muss stets die Reihenfolge der Operationen beachtet werden: Die allgemeine Nichtkommutativität der Matrixmultiplikation führt dazu, dass eine Folge zweier affiner Transformationen je nach Reihenfolge zu unterschiedlichen Ergebnissen führen kann (siehe Abbildung 4.3.1). Ebenso kann ein auf das Bild gelegtes Rauschsignal durch eine folgende Glättungsoperation wieder verschwinden (siehe Unterabschnitt 4.3.2). Details zu einzelnen Bildbearbeitungsverfahren sind der Fachliteratur zu entnehmen.

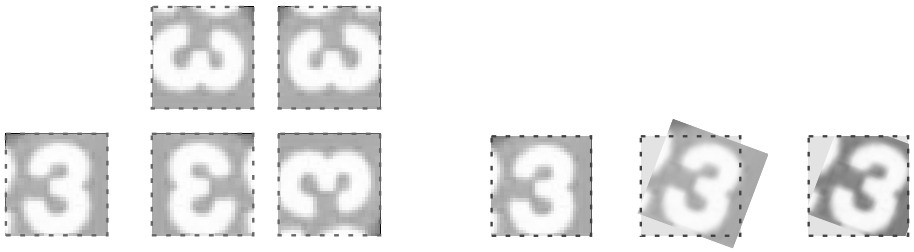
4.3.1 *Geometrische Augmentationen*

In geometrischen Augmentationsverfahren werden die Positionen der einzelnen Pixel verändert [WGSM16]. Dies kann durch affine oder elastische Transformationen geschehen. Affine Transformationen ordnen einem n-dimensionalen Vektor $\mathbf{x}_e$ über eine lineare Abbildung einen neuen Positionsvektor $\mathbf{x}_a$ zu. Im Falle der zweidimensionalen Bilder ist $n = 2$. Die Transformation lässt sich wie folgt beschreiben [NFHS12], S.241):

$$\mathbf{x}_a = \mathbf{A}\mathbf{x}_e + \mathbf{v} \tag{4.3.1}$$

$\mathbf{A}$ ist hierbei eine 2×2 Koeffizientenmatrix. $\mathbf{A}$ stellt eine lineare Transformation dar; die Koeffizienten bestimmen die Art der Transformation. Durch verschiedene Koeffizienten lassen sich u. a. Skalierungen, Rotationen, Scherungen und Spiegelungen beschreiben. $\mathbf{v}$ gibt die Verschiebung der Pixel an, welche über das gesamte Bild konstant bleibt. Genaueres zu affinen Transformationen lässt sich in [NFHS12], Kapitel 9.4 nachlesen.
Um das augmentierte Bild zu erzeugen wird eine neue Bildmatrix erstellt. Die Elemente der neuen Bildmatrix werden dabei gemäß der Transformation aus dem Originalbild abgetastet. Für bestimmte Positionen im Bild müssen ggf. die umliegenden Pixelwerte interpoliert werden, wozu verschiedene Interpolationsverfahren zu Verfügung stehen. Details hierzu finden sich in der Fachliteratur. Die Auflösung der Bilder wird bei den Augmentationen beibehalten. Bereiche des Originalbildes, die bei der Transformation außerhalb der neuen Bildgrenzen verschoben werden, gehen unwiderruflich verloren. Zusätzlich kann es vorkommen, dass im augmentierten Bild Bereiche vorkommen, die nicht im Originalbild zu finden waren. In diesem Fall müssen die entsprechenden Pixelbereiche aufgefüllt werden, wozu ebenfalls mehrere Verfahren eingesetzt werden:

- Konstanten Helligkeitswert festlegen.
- Farbe des nächsten Pixel im Originalbild verwenden.
- Originalbild an der Kante spiegeln und Pixel abtasten.
- Originalbild kacheln und Pixel abtasten.



(a) (b)

Abbildung 4.3.1: (a): Affine Transformationen sind nicht kommutativ. Das Bild der 3 wird horizontal gespiegelt und um 90° gekippt. Die Vertauschung der Reihenfolge der Operationen führt zu unterschiedlichen Ergebnissen. (b): Demonstration einer Rotationsaugmentation. Man sieht, wie Bereiche des Bildes abgeschnitten und andere Bereiche mit einem Grauwert aufgefüllt werden.

Zusätzlich zu affinen Transformationen existieren auch elatische Transformationen, die ein Bild durch lokale Verschiebungsfelder (*displacement fields*) verzerren [SSP+03]. Auch andere nichtlineare Transformationen sind möglich. In dieser Arbeit werden verschiedene affine Transformationen verwendet. Eine Übersicht der genutzten Transformationen ist in Tabelle 4.3.1 zu sehen.

Verfahren	Parameter	Beispielbilder
Verschiebung	Stärke in Pixeln, x und y	
Skalierung	Skalierungsfaktor für x und y	
Rotation	Winkel α	
Horizontale Scherung	Winkel α	

Tabelle 4.3.1: Die betrachteten affinen Transformationen, angewendet auf ein Beispielbild. Mit Kombinationen dieser Transformationen werden die Bilder augmentiert.

4.3.2 *Fotometrische Augmentationen*

Fotometrische Verfahren verändern den Helligkeitswert einzelner Pixel. Der neue Farb- oder Helligkeitswert eines Pixels entspricht dem Ergebnis einer Abbildung, die ggf. vom ursprünglichen Farbwert des Pixels oder seiner Umgebung abhängig ist. Manche der Operationen werden durch eine zweidimensionale diskrete Signalfaltung durchgeführt. Näheres zu Filteroperationen findet sich in [NFHS12], Kapitel 5.2. Es muss sichergestellt werden, dass die Pixeldaten der Bilder nach einer arithmetischen Operation auf den Wertebereich $[0, 1]$ normiert werden. Dazu werden zu große oder zu geringe Pixelwerte auf die Randwerte gesetzt (*clipping*). Stattdessen lässt sich das Bild auch wieder in den gültigen Wertebereich skalieren (vgl. [NFHS12], S.90). Es werden hierbei die folgenden fotometrischen Augmentationen untersucht:

- **Inversion**
  Der Helligkeitswert eines Pixels x wird auf $1 - x$ abgebildet, wodurch das Negativ des Bildes entsteht. Diese Augmentation wird nur auf einen Prozentsatz der Bilder angewendet.

Abbildung 4.3.2: Ziffernbilder und ihre zugehörigen Negative.

- **Rauschen**
  Das Bild wird pixelweise mit einem zufällig generierten Rauschsignal behaftet, welches auf die Bildpixel entweder addiert oder multipliziert wird. Für die Generierung des Rauschsignals muss eine passende Wahrscheinlichkeitsverteilung für die Verteilung der Werte des Rauschsignals gewählt werden. Hierzu wird die *Gaußsche Normalverteilung* verwendet. Die Dichtefunktion ist wie folgt gegeben ([Mur12], S.38):

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.3.2}$$

  Der Parameter μ ist der Medianwert der Verteilung. Um ein symmetrisches Rauschsignal zu erzeugen wird dieser im additiven Fall auf 0 und im multiplikativen Fall auf 1 gesetzt. Der Parameter σ beschreibt die Standardabweichung.

Die Variation des Parameters verstärkt oder mindert den Effekt des Rauschsignals.

Abbildung 4.3.3: Auf ein Testbild wurde ein durch eine Gaußsche Normalverteilung gegebenes Rauschen addiert. Der Parameter σ der Standardabweichung wurde kontinuierlich erhöht.

- **Unschärfe**
  Auf das Bild wird eine Glättungsoperation angewandt, die harte Kanten glättet und feine Details verschwimmen lässt. Auf diesem Weg kann ein Bild von störenden Rauschelementen befreit werden, ein zu starkes Weichzeichnen lässt jedoch die Zahl selbst unkenntlich werden. Eine Bildglättung kann durch eine diskrete Faltungsoperation realisiert werden. Einige passende Filtermasken hierfür werden in [GW02], S.119ff sowie [NFHS12], Kapitel 5.2 und 5.3 aufgeführt und erklärt. Das Weichzeichnen eines Bildes lässt sich als Tiefpassfilter im Frequenzbereich betrachten ([GW02], S.167).
  In den Versuchen wird der *Gaußsche Weichzeichner* verwendet. Hierzu wird eine zweidimensionale Gaußsche Normalverteilung erzeugt (vgl. Gleichung 4.3.2). Aus dieser werden um den Ursprung diskret Werte abgetastet und in eine Matrix eingetragen, welche darauf als Filterkern für die Glättung verwendet wird. Die Standardabweichung σ ist hier wählbar und hat Einfluss auf die Stärke des Weichzeichners. Näheres zum Gaußfilter lässt sich in [Pri15], Kapitel 6.2.6 nachlesen.

Abbildung 4.3.4: Das Testbild wurde kontinuierlich stärker mit einem Gaußschen Weichzeichner bearbeitet. Der Parameter σ wird stetig erhöht.

- **Helligkeits- und Kontrastanpassung** Durch das Addieren oder Multiplizieren der Bildpixel mit einem konstanten Wert wird die Helligkeit des Bildes verändert, wodurch ein Bild erhellt oder verdunkelt werden kann. Dies führt zu veränderten Kontrastverhältnissen, die einen Einfluss auf die Gradientenbilder

haben. Der Kontrast eines Bildes lässt sich um den Faktor α durch folgende Operation linear skalieren:

$$x_e = \alpha(x_a - 0{,}5) + 0{,}5 \tag{4.3.3}$$

Ein α größer als 1 erhöht den Kontrast, wodurch helle Stellen des Bildes noch stärker erhellt werden, und dunkle Stellen noch dunkler werden. Ist α kleiner gleich 0, so vermindert sich der Kontrast, wodurch das Bild mehr und mehr sich dem konstanten Grauwert für 0,5 annähert.

Abbildung 4.3.5: Helligkeits- und Kontrastveränderungen auf dem Bild. Oben durch ein additives Verfahren, mittig durch Multiplikation. Unten: Kontrastanpassung durch Variationen von α. In der Mitte ist stets das unveränderte Originalbild.

5 EVALUIERUNG

In diesem Kapitel werden die Ergebnisse des vorherigen Kapitels vorgestellt. Wie in Kapitel 4 beschrieben, wird zunächst ein Klassifikator mit den Bildern aus SVHN TRAIN trainiert. Sowohl für die HOG-Merkmalsrepräsentation der Bilder wie auch für die beiden Klassifikatoren werden die optimalen Hyperparameter in einer Kreuzvalidierung bestimmt. In Abschnitt 5.2 werden die optimalen Parameterkonfigurationen für den Klassifikator beschrieben. Der Einfluss der einzelnen Hyperparameter wird gezeigt, und es werden die Support Vector Machine und der KNN-Klassifikator hinsichtlich ihrer Klassifikationsleistung verglichen. Die so ermittelten Parameterkonfigurationen für die Klassifikatoren und das HOG-Verfahren werden darauf auf SVHN TEST evaluiert, und die Ergebnisse aufgelistet. Zur Evaluation werden mehrere Metriken verwendet, die in Abschnitt 5.1 erläutert werden.
In Abschnitt 5.4 wird darauf der Einfluss der Erweiterung der Trainingsmenge auf die Klassifikationsleistung gezeigt. Es wird untersucht, wie sich die in Abschnitt 4.3 vorgestellten Augmentationsverfahren auf die Klassifikationsleistung auswirken. Zusätzlich werden die Zusammensetzung und die Gesamtgröße der Trainingsmenge betrachtet, und auf SVHN TEST evaluiert.

5.1 EVALUATIONSMETRIKEN

Zur Evaluation eines Klassifikators wird von diesem eine Menge von annotierten Testobjekten klassifiziert. Für jedes Testobjekt wird die ermittelte Klasse mit der tatsächlichen Klasse verglichen. Stimmen diese überein, liegt der Klassifikator richtig, falls nicht, liegt eine Fehlentscheidung vor. Um die Erkennungsleistung eines Klassifikators zu messen werden verschiedene *Evaluationsmetriken* verwendet, welche entweder über die gesamte Trainingsmenge, oder klassenweise erhoben werden. Die einfachste betrachtete Metrik ist die Erkennungsrate, welche sich aus dem Verhältnis der korrekt klassifizierten Objekten zu allen vorhandenen Testobjekten berechnet. Weiterhin werden in der Mustererkennung die binären Maße *Precision*, *Recall* und der *F1-Score* verwendet. Diese werden klassenweise ermittelt, und geben Einblick über die Erkennungsleistung für die Objekte einer einzelnen Klasse. Precision, Recall und F1-Score werden in Unterabschnitt 5.1.1 näher erläutert. Ein weiteres wichtiges Maß ist die

Konfusionsmatrix, die in Unterabschnitt 5.1.2 beschrieben wird.
Diese Maße befassen sich alle mit der Erkennungsleistung des Klassifikators. In der Praxis können Laufzeit und Speicherbedarf für das Trainieren oder Klassifizieren ebenfalls von enormer Bedeutung sein (siehe [DHS01], S.113).

5.1.1 *Precision, Recall, F1-Score*

Precision, Recall und der F1-Score sind drei Maße, die in binären Entscheidungssituationen verwendet werden. Da bei der Hausnummernerkennung insgesamt 10 Klassen vorliegen, werden diese Maße immer klassenweise erhoben. Für eine Klasse K_i liegt für ein Objekt eine *positive* Entscheidung vor, falls der Klassifikator das Objekt der Klasse K_i zuordnet, und eine *negative* Entscheidung, falls der Klassifikator das Objekt stattdessen einer anderen Klasse $K_{j,\, j\neq i}$ zuordnet. Der Klassifikator kann ein Objekt richtig oder fehlerhaft einstufen, wodurch vier Fälle entstehen, welche in Tabelle 5.1.1 aufgeführt sind.

		Tatsächliches Klassenlabel	
		i	$j,\ j \neq i$
Geschätztes Klassenlabel	i	*True positive* (TP)	*False positive* (FP)
	$j,\ j \neq i$	*False negative* (FN)	*True negative* (TN)

Tabelle 5.1.1: Die vier möglichen Entscheidungsfälle, die bei der Klassifikation eines Objekts entstehen (siehe [Mur12], S.181).

Seien TP_i, TN_i, FP_i und FN_i die Mengen, die jeweils die richtig positiv, richtig negativ, falsch positiv und die falsch negativ eingestuften Objekte für eine Klasse K_i beschreiben. Aus diesen Mengen lassen sich die folgenden Mengen berechnen (siehe [Mur12], S.181):

- Die tatsächlich zur Klasse K_i gehörenden Objekte:

$$N_i^+ = TP_i + FN_i \tag{5.1.1}$$

- Alle nicht zur Klasse K_i gehörenden Objekte:

$$N_i^- = FP_i + TN_i \tag{5.1.2}$$

- Alle von dem Klassifikator in die Klasse K_i eingeordneten Objekte:

$$\hat{N}_i^+ = TP_i + FP_i \tag{5.1.3}$$

- Alle von dem Klassifikator nicht in die Klasse K_i eingeordneten Objekte:

$$\hat{N}_i^- = FN_i + TN_i \tag{5.1.4}$$

Für eine Klasse K_i lässt sich die Precision P_i definieren. Die Precision gibt für die Klasse an, wie viele der von dem Klassifikator zu der Klasse gerechneten Objekte tatsächlich zu dieser Klasse gehören. Werden also viele Ziffern fälschlicherweise einer bestimmten Klasse zugeordnet, so ist für diese Klasse die Precision sehr gering. Werden der Klasse hingegen fast ausschließlich nur zugehörige Objekte zugeordnet, so ist die Precision sehr hoch. Mathematisch definiert sich die Precision wie folgt (vgl. [Mur12], S.182):

$$P_i = \frac{|TP_i|}{|TP_i \cup FP_i|} = \frac{|TP_i|}{\left|\hat{N}_i^-\right|} \tag{5.1.5}$$

Der Recall R_i gibt hingegen für die Klasse K_i an, wie viele der Objekte x mit dem Klassenlabel i der Klassifikator in der Klasse aufgenommen hat. Bei einem niedrigen Recall existieren also für eine Klasse viele zugehörige Objekte, die zu anderen Klassen gerechnet wurden. Ist der Recall hingegen hoch zeigt dies, dass die meisten Objekte der Klasse richtig klassifiziert wurden. Der Recall berechnet sich wie folgt (vgl. [Mur12], S.182):

$$R_i = \frac{|TP_i|}{|TP_i \cup FN_i|} = \frac{|TP_i|}{\left|N_i^+\right|} \tag{5.1.6}$$

Zusätzlich wird häufig auch der F1-Score verwendet, welcher sich aus dem harmonischen Mittel von Precision und Recall zusammensetzt (siehe [Mur12], Abschnitt 5.7.2.3):

$$F_i^1 = \frac{2}{1/P_i + 1/R_i} = \frac{2P_iR_i}{R_i + P_i}$$

(5.1.7)

Diese Metriken können verwendet werden, um die Leistung des Klassifikators für jede Klasse einzeln zu beurteilen. Sie bieten einen tiefen Einblick in die Arbeitsweise des Klassifikators, und erlauben es, besonders schwierige Klassen zu identifizieren. Ein noch feineres Werkzeug zur Ermittlung der Leistung des Klassifikators ist die *Konfusionsmatrix*, die im Folgenden vorgestellt wird.

5.1.2 *Konfusionsmatrix*

Werden n verschiedene Klassen betrachtet ist die Konfusionsmatrix A eine $n \times n$ Matrix, welche die Verwechslungsraten zwischen einzelnen Klassen anzeigt. Ein Element $A_{i,j}$ der Matrix zeigt die Anzahl der Elemente der Klasse K_i an, die zu der Klasse K_j eingestuft wurden ([TK08], S.573). Für $i = j$ zeigt die Konfusionsmatrix die Anzahl der richtig klassifizierten Elemente an. Hat ein $A_{i,j,\, i \neq j}$ also einen unüblich hohen Wert zeigt dies, dass Elemente der Klasse K_i häufig als Objekte von K_j fehlklassifiziert werden. Die Klassen werden also häufig miteinander verwechselt.
Aus der Konfusionsmatrix lassen sich direkt Precision und Recall für eine Klasse bestimmen. Für eine Klasse K_i und einer Matrix A können Precision P_i und Recall R_i wie folgt bestimmt werden (vgl. [TK08], S.573):

$$P_i = \frac{A_{i,i}}{\sum_n A_{n,i}} \tag{5.1.8a}$$

$$R_i = \frac{A_{i,i}}{\sum_n A_{i,n}} \tag{5.1.8b}$$

Eine Konfusionsmatrix kann absolut oder relativ gewertet werden. Im absoluten Fall werden in der Konfusionsmatrix alle klassifizierten Objekte gezählt. Im relativen Fall wird die Konfusionsmatrix zeilenweise oder spaltenweise normiert. Werden die Spalten der Konfusionsmatrix normiert, so summieren sich alle Spalten der Matrix zu 1. Damit lässt sich die Präzision gut beobachten. Bei einer zeilenweisen Normierung der Zeilen kann hingegen der Recall einer Klasse visuell anschaulich dargestellt werden.

5.2 PARAMETERAUSWERTUNG

Das Ziel der Versuche ist es, im ersten Schritt eine optimale Parameterkonfiguration für den Klassifikator und die HOG-Merkmalsvektoren zu ermitteln. Es soll zusätzlich bestimmt werden, ob der KNN-Klassifikator oder die Support Vector Machine für diese Klassifikationsaufgabe besser geeignet sind. Um overfitting (siehe Abschnitt 2.1) zu vermeiden, werden die Parameterkombinationen mit einer Kreuzvalidierung mit 5 Folds geprüft. Für jeden Fold wird die Erkennungsrate ermittelt. Das Ergebnis eines Versuchsdurchlaufes ist das arithmetische Mittel aller 5 Folds der Kreuzvalidierung. Basierend auf den Ergebnissen der Kreuzvalidierung lassen sich die optimalen Hyperparameter ermitteln.

5.2.1 *HOG-Deskriptoren*

Wie in [DT05] und Abschnitt 2.2 beschrieben, hängt besonders das HOG-Verfahren von einer großen Menge von Parametern ab. Für die HOG-Merkmalsdeskriptoren werden die folgenden Parameter geprüft:

- **Gammakorrektur**:
  Auf dem Bild kann zunächst eine Gammakorrektur vorgenommen werden, indem die Helligkeitswerte des Bildes auf ihre Quadratwurzel abgebildet werden: $x \to \sqrt{x}$.
- **Größe der Gradientenhistogramme**:
  Dieser Parameter gibt die Anzahl der Werte in den Gradientenhistogrammen an. Je größer dieser gewählt wird, desto feiner ist die Verteilung der Richtungsvektoren im Histogramm.
- **Zellengröße in Pixeln**:
  Die Größe der Zellen in Pixeln.
- **Blockgröße**:
  Dieser Parameter gibt an, wie viele Zellen in jedem HOG-Deskriptorblock enthalten sind. Für die Evaluation werden R-HOGs verwendet.
- **Normierungsfunktion**:
  Das Normierungsverfahren wird aus einer der vier in Abbildung 2.2.1 aufgeführten Verfahren gewählt.

Als Basiskonfiguration wird eine an die Ergebnisse von [DT05] angelehnte Konfiguration genutzt, welche im Folgenden beschrieben wird. Wie in [DT05] empfohlen, werden zunächst 9 Einträge in den Gradientenhistogrammen verwendet. Die Zellen sind anfangs 8×8 Pixel groß, und die Blöcke jeweils 2×2 Zellen. Als Normierungsverfahren wird die L_2-Norm verwendet, und auf eine Gammakorrektur wird verzichtet. Die Parameter wurden schrittweise ermittelt. Zum Klassifizieren wurde zunächst eine Support Vector Machine mit einem C-Parameter von 0,05 genutzt. Als Mehrklassenstrategie wurde das 1 gegen 1 Verfahren gewählt. Die Parameter des Klassifikators werden in Unterabschnitt 5.2.2 optimiert. Alle Ergebnisse wurden auf zwei Nachkommastellen gerundet.

Block- und Zellengröße

Als erstes wird die Größe der HOG-Deskriptoren analysiert. Da die Ausgangsbilder stets 32×32 Pixel groß sind bietet es sich an, als Zellengrößen Teiler von 32 zu nehmen. Untersucht werden daher Zellen von 4×4, 8×8 und 16×16 Pixeln. Als Blockgröße werden entweder einzelne Zellen verwendet (1×1) oder 2×2-Blöcke. Die Ergebnisse sind in Tabelle 5.2.1 aufgelistet.

		Blockgröße	
		1×1	2×2
Zellengröße	4×4	83,07%	**85,35%**
	8×8	78,36%	79,74%
	16×16	54,15%	51,84%

Tabelle 5.2.1: Klassifikationsergebnisse für die Kreuzvalidierung für verschiedene Block- und Zellgrößen der HOG-Deskriptoren.

Es ist deutlich zu sehen, dass mit kleineren Zellengrößen bessere Ergebnisse erreicht werden, als mit größeren Zellen. Eine zellenübergreifende Blocknormierung sollte zudem vorgenommen werden, da dies ebenfalls die Klassifikationsleistung verbessert.

Weiterhin wurde die optimale Anzahl an Werten in den Zellenhistogrammen untersucht. Ausgehend von den zuvor ermittelten Zellen- und Blockgrößen wurden von 6 bis 12 alle Histogrammwerte untersucht. Es zeigt sich, dass ein Histogramm mit 9 Einträgen mit einer Erkennungsrate von 85,35% die besten Ergebnisse erzielt. Die Ergebnisse werden in Tabelle 5.2.2 gezeigt.

	Größe des Histogramms						
	6	7	8	9	10	11	12
Erkennungsrate	84,89%	85,02%	85,25%	**85,35**%	85,25%	85,22%	85,16%

Tabelle 5.2.2: Erkennungsraten für die HOG-Deskriptoren bei unterschiedlichen Histogrammgrößen.

Normierungsverfahren und Gammakorrektur

Die vier von Dalal und Triggs untersuchten Normierungsfunktionen wurden evaluiert. Es stellt sich die L_2-Norm als die beste der vier Methoden heraus. Die Klassifikationsraten für die Normierungsverfahren werden in Tabelle 5.2.3 aufgezeigt.

	Norm			
	L1	L1-sqrt	L2	L2-Hys
Erkennungsrate	84,48%	84,53%	**85,35**%	73,20%

Tabelle 5.2.3: Klassifikationsergebnisse für die vier vorgestellten Normierungsverfahren bei der Berechnung der HOG-Deskriptoren.

Auffällig schlecht schneidet hierbei das L2-Hys Verfahren mit 73,20% Erkennungsrate ab. Es ist anzunehmen, dass beim Einschränken des Wertebereichs des Gradientenhistogramms auf $[0, 0{,}2]$ zu viele Informationen verloren gehen.
Zuletzt wurde der Einfluss der Gammakorrektur $x \to \sqrt{x}$ auf die HOG-Deskriptoren untersucht. Diese ist weitgehend ineffektiv, und mindert die Erkennungsrate in der Kreuzvalidierung von 85,35% auf 85,17%.
Mit der Ausnahme weniger Fälle, wie die L2-Hys Norm als Normierungsverfahren, bewegen sich die Klassifikationsraten für das HOG-Verfahren nur im Bereich weniger Prozentpunkte.

5.2.2 *Klassifikatoren*

Es gilt, aus der Support Vector Machine und dem KNN-Klassifikator den besten Klassifikator zu ermitteln. Beide Klassifikatoren sind, wie das HOG-Verfahren, von einer Anzahl von Hyperparametern abhängig, die einen Einfluss auf die Klassifikationsleistung nehmen. Zur Evaluation werden die im vorherigen Schritt bestimmten HOG-

Deskriptoren verwendet. Je nach dem gewählten Klassifikator werden unterschiedliche Parameter betrachtet.

KNN-Klassifikator

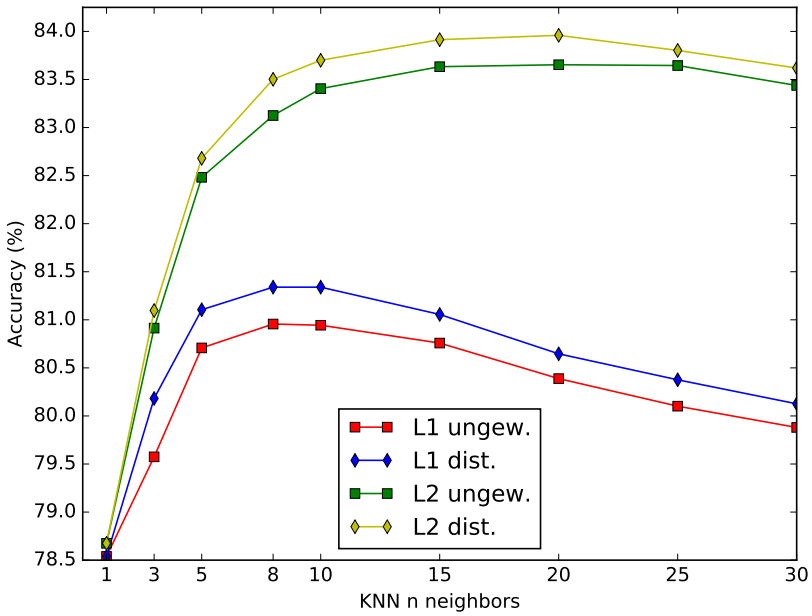


Abbildung 5.2.1: Die Erkennungsleistungen für verschiedene Parameter des KNN-Klassifikators.

Es werden zunächst die Ergebnisse der Kreuzvalidierung für den KNN-Klassifikator vorgestellt. Die folgenden Parameter werden evaluiert:

- **Nachbaranzahl K**:
  Die gesamte Anzahl der Nachbarn, die für jedes Sample betrachtet werden. Betrachtet wurden die Werte $k = 1, 3, 5, 8, 10, 15, 20, 25, 30$.
- **Distanzmaß**:
  Für die Klassifikation wurden der Cityblock-Abstand L_1 und die euklidische Distanz L_2 betrachtet (siehe Unterabschnitt 4.2.1).
- **Gewichtung der Nachbarn**:
  Die k nächsten Nachbarn werden entweder ungewichtet gewertet, oder es wird die inverse Distanz zu dem Objekt als Gewicht gewählt.

Für jedes betrachtete k werden also die vier Kombinationen von Distanzmaß und Nachbargewichtung betrachtet. Die Ergebnisse werden in Abbildung 5.2.1 dargestellt. Aus den ermittelten Erkennungsraten geht hervor, dass für alle k das euklidische Abstandsmaß bessere Ergebnisse erzielt als der Cityblock-Abstand. Zusätzlich wird die Erkennungsleistung durch eine Distanzgewichtung erhöht. Die höchste Erkennungsrate liegt für $k = 20$ vor und beträgt 83,6%.

Support Vector Machine

Für die Support Vector Machine werden die folgenden Parameter evaluiert:

- **Regularisierung C**:
  Die Bedeutung des Regularisierungsparameters C wird in Unterabschnitt 2.4.3 erläutert. Der Parameter gibt an, wie streng die Randbedingung der Hyperebene eingehalten wird.

- **Mehrklassenstrategie**:
  Es wird evaluiert, ob eine 1 gegen 1, oder eine 1 gegen n SVM eine bessere Erkennungsleistung bietet. Einzelheiten werden in Unterabschnitt 2.4.4 erläutert.

Die Ergebnisse für die Parameterevaluation der SVM sind in Abbildung 5.2.2 zu sehen. Die besten Ergebnisse können mit der 1 gegen 1-Strategie erreicht werden. Für diese Strategie liefert ein C von 0,0375 die besten Klassifikationsergebnisse, mit einer Erkennungsrate von 85,40%. Die 1 gegen n-Strategie kann mit einem Höchstwert von 84,27% Erkennungsrate nicht mit der 1 gegen 1-Strategie mithalten.

5.3 KLASSIFIKATIONSERGEBNISSE

Aus der vorhergegangenen Parameterevaluation gehen folgende Ergebnisse hervor: Für den KNN-Klassifikator liefert ein k von 20 die besten Ergebnisse. Die euklidische Norm übertrifft stets den Cityblock-Abstand. Somit ist für den KNN-Klassifikator $k = 20$, unter verwendet der L_2-Norm und gewichteten Abständen die beste Wahl. Die lineare Support Vector Machine kann in jedem Fall den KNN-Klassifikator in den Ergebnissen übertreffen. Als Mehrklassenstrategie sollte 1 gegen 1 verwendet werden. Ein C von 0,0375 liefert die besten Ergebnisse.
Für die HOG-Deskriptoren haben 4×4 Pixel große Zellen mit einer Blockgröße von 2×2 die besten Ergebnisse geliefert. Als Histogrammgröße sollte 9 gewählt werden, als Normierungsfunktion die L2 Norm. Auf eine Gammakorrektur vor der Klassifikation kann verzichtet werden.

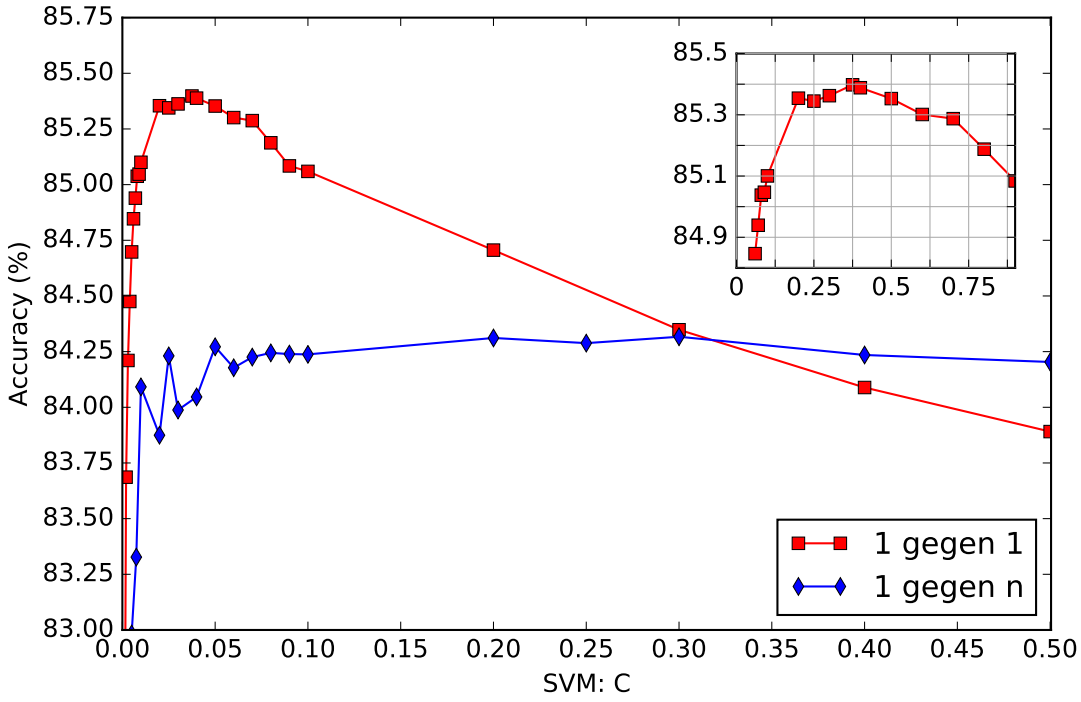


Abbildung 5.2.2: Die Klassifikationsleistungen für die lineare SVM in der Kreuzvalidierung. Rot ist die 1 gegen 1-Strategie, Blau die 1 gegen n-Strategie. Bei $C = 0{,}0375$ sind für die 1 gegen 1-Strategie die besten Ergebnisse sichtbar.

In der Kreuzvalidierung kann mit dieser Konfiguration eine Erkennungsrate von 85,40% erreicht werden. Als nächstes gilt es, diese Parameterkonfiguration auf der Testmenge SVHN TEST zu evaluieren.

Es wird ein Klassifikator mit der zuvor ermittelten Parameterkonfiguration auf allen Elementen von SVHN TRAIN trainiert. Nachdem der Klassifikator trainiert wurde, wird dieser auf SVHN TEST evaluiert. Es werden Statistiken über Precision, Recall und den F1-Score sowie die Konfusionsmatrix erhoben. Die Konfusionsmatrix in Abbildung 5.3.1 abgebildet. Für jede Klasse sind in Tabelle 5.3.1 Precision, Recall und F1-Score aufgelistet.

Aus der Konfusionsmatrix in Tabelle 5.3.1 geht eine Klassifikationsrate von insgesamt 84,60% hervor. Der nächste Schritt ist die Erweiterung der Trainingsmenge durch augmentierte Bilder, sowie Bilder aus SVHN EXTRA. Die Ergebnisse dieses Schrittes werden im folgenden Abschnitt beschrieben.

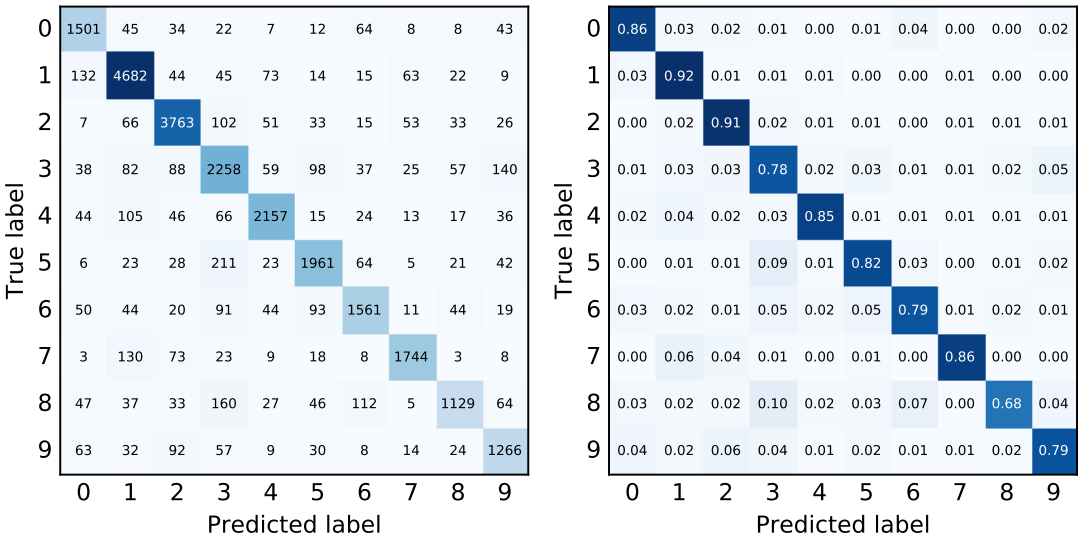


Abbildung 5.3.1: Die Konfusionsmatrizen für den mit SVHN TRAIN trainierten Klassifikator. Links die absoluten Zahlen, rechts spaltenweise normiert.

5.4 ERWEITERTE TRAININGSMENGE

Um die Leistung des Klassifikators weiter zu verbessern wird nun die Trainingsmenge um Bilder aus SVHN EXTRA sowie augmentierte Bilder erweitert. Zunächst wird also nach den gewählten Parametern, wie in Unterabschnitt 4.2.2 beschrieben, eine Trainingsmenge konstruiert. Es wird die in Abschnitt 5.2 ermittelte Parameterkombination für die HOG-Deskriptoren und den Klassifikator verwendet. Der erzeugte Klassifikator wird direkt auf SVHN TEST evaluiert. Es gilt also, zum Trainieren eines Klassifikators eine Trainingsmenge zu erzeugen, mit der eine möglichst gute Erkennungsleistung erreicht werden kann.

5.4.1 *Augmentationsverfahren*

Für eine erfolgreiche Augmentierung der Trainingsmenge ist eine geschickte Auswahl der Augmentationsverfahren von enormer Bedeutung. In Abschnitt 4.3 wurden einige geometrische und fotometrische Verfahren vorgestellt, welche für die Experimente evaluiert wurden. Es reicht nicht aus, willkürlich Augmentationen durchzuführen: Ungünstige Augmentationsverfahren, die das Bild zu sehr verfälschen oder undeutlich machen, mindern die Klassifikationsleistung. Empirisch hat sich eine Reihe von

Klasse	Precision	Recall	F1-Score
0	79,38%	86,07%	82,59%
1	89,25%	91,82%	90,52%
2	89,15%	90,70%	89,92%
3	74,40%	78,35%	76,32%
4	87,72%	85,49%	86,59%
5	84,53%	82,26%	83,38%
6	81,81%	78,96%	80,36%
7	89,85%	86,83%	88,08%
8	83,14%	68,01%	74,82%
9	76,59%	79,37%	77,96%
⌀, ungew.	83,58%	82,74%	83,05%
⌀, gew.	84,66%	84,60%	84,55%

Tabelle 5.3.1: Precision, Recall und der F1-Score für jede Klasse. Unten der ungewichtete Durchschnittswert aller Messwerte, darunter werden die selben Werte durch die Klassengrößen der Testmenge gewichtet.

Augmentationsverfahren als besonders vielversprechend herausgestellt. Jedes der folgenden Augmentationsverfahren wird zu einem unterschiedlichen Grad auf das Bild angewendet. Folglich ist es zusätzlich notwendig zu bestimmen, wie stark sich eine Augmentation auf das Bild auswirken soll. Beispielsweise muss bei einer Rotation des Bildes ein Winkel bestimmt werden, um den das Bild rotiert wird. Hierzu wird empirisch ein Wertebereich ermittelt, in welchem die Parameter einer Augmentation liegen können. Bei der Augmentation eines Bildes wird dann zufällig aus dem ermittelten Wertebereich ein Wert bestimmt, der verwendet wird. Die Augmentationsverfahren, sowie ihre verwendeten Wertebereiche werden im Folgenden aufgelistet:

- **Affine Transformationen**:
  Affine Transformationen sind sehr effektiv, um ein Bild zu verändern. Ein Bild wird um seinen Mittelpunkt rotiert, skaliert, und horizontal geschert. Darauf wird das Bild um bestimmte Anzahl von Pixeln horizontal und vertikal verschoben. Beim Durchführen der Transformationen entstehen im Bildbereich leere Bereiche, die zu keinem Ursprungspixel zugeordnet werden können. Zur Füllung der Bereiche mit Pixelwerten werden die in Abschnitt 4.3 beschriebenen

Verfahren verwendet. Die optimalen Wertebereiche für die Parameter sind in Tabelle 5.4.1 angegeben. Für jedes Bild werden alle Parameter unabhängig voneinander neu bestimmt.

Verschiebung in Prozent		Skalierung in Prozent		Rotation	Scherung
x	y	x	y	α	α
$[-0{,}1,\ 0{,}1]$	$[-0{,}1,\ 0{,}1]$	$[0{,}8,\ 1{,}2]$	$[0{,}7,\ 1{,}2]$	$[-10°,\ 10°]$	$[-10°,\ 10°]$

Tabelle 5.4.1: Die genutzten affinen Transformationen und die betrachteten Wertebereiche ihrer Parameter.

- **Inversion**:
  Es wird mit einer bestimmten Wahrscheinlichkeit ein Bild auf sein Negativ abgebildet. Der optimale ermittelte Wahrscheinlichkeitswert ist 0,25. Von allen augmentierten Bildern wird also ein Viertel invertiert.

- **Kontrastnormierung**:
  Für jedes Bild wird eine Kontrastnormierung vorgenommen. Der zugehörige Parameter α wird zufällig aus dem Wertebereich von $[0{,}8,\ 1{,}2]$ gewählt. Für jedes Bild wird der Kontrast also leicht erhöht oder gemindert.

- **Gaußscher Weichzeichner**:
  Auf jedes Bild wird ein Gaußscher Unschärfefilter gelegt, wodurch jedes Bild in verschiedenen Maßen unscharf gemacht wird. Dazu wird aus einer Gaußschen Normalverteilung eine Filtermaske erstellt, mit der die Kanten aus dem Bild verschwinden (siehe Abschnitt 4.3). Das zugehörige σ der Standardabweichung wird zufällig aus dem Intervall $[0,\ 2]$ bestimmt.

- **Additives Gaußsches Rauschen**:
  Es wurde versucht, auf die Bilder ein Rauschsignal zu addieren. Dieses Verfahren hat jedoch keine erfolgreichen Resultate gezeigt, weshalb es im weiteren nicht verwendet wird.

Diese Augmentationen werden in der genannten Reihenfolge auf das Bild angewendet. Einige Beispielbilder und die daraus generierten augmentierten Bilder sind in Abbildung 5.4.1 zu sehen.

5.4.2 *Komposition der Trainingsmenge*

Die in Unterabschnitt 5.4.1 bestimmten Augmentationsverfahren werden verwendet, um die Trainingsmenge zu erweitern. Neben den Augmentationsverfahren stehen zusätzlich in SVHN EXTRA eine Vielzahl zusätzlicher Trainingsbilder zur Verfügung. Es sollen so viele Bilder wie möglich durch Augmentationen erzeugt werden. Wie in Unterabschnitt 4.2.2 beschrieben wurden hierzu mehrere Konfigurationen aus Trainingsbildern betrachtet. Es werden n Bilder generiert, von denen entweder alle, die Hälfte, oder ein Viertel der Bilder aus dem SVHN-Datensatz stammen. Die verbleibenden Bilder werden durch Augmentationen erzeugt, wobei bei den augmentierten Trainingsmengen für jedes Bild jeweils ein oder drei augmentierte Bilder generiert werden. Die Zusammensetzungen werden in Abbildung 5.4.2 dargestellt.

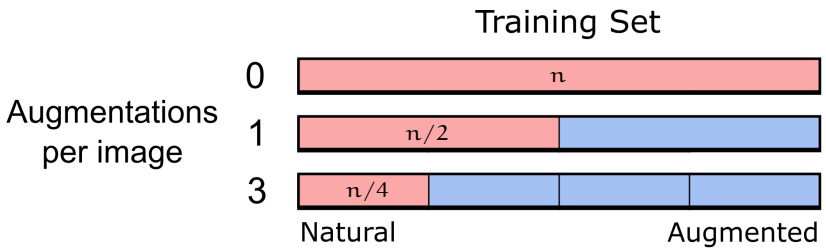


Abbildung 5.4.2: Die drei möglichen Zusammensetzungen für die augmentierte Trainingsmenge für ein festgelegtes n.

Um diese Zusammensetzungen zu evaluieren wurden Trainingsmengen für verschiedene Gesamtgrößen n erzeugt. Dafür wurde die benötigte Anzahl von natürli-

Abbildung 5.4.1: Zufällig ausgesuchte Beispiele für die augmentierten Bilder. Links ist stets das Originalbild sichtbar, rechts davon je fünf augmentierte Varianten. Die Ursprungsbilder wurden zufällig aus SVHN TRAIN und SVHN EXTRA entnommen.

chen Bildern aus SVHN TRAIN und SVHN EXTRA genutzt und ggf. ein- oder mehrmals augmentiert. Zur Klassifikation wurde der in Abschnitt 5.3 bestimmte Klassifikator, sowie die zugehörige Konfiguration der HOG-Deskriptoren verwendet. Die Ergebnisse werden in Abbildung 5.4.3 gezeigt.

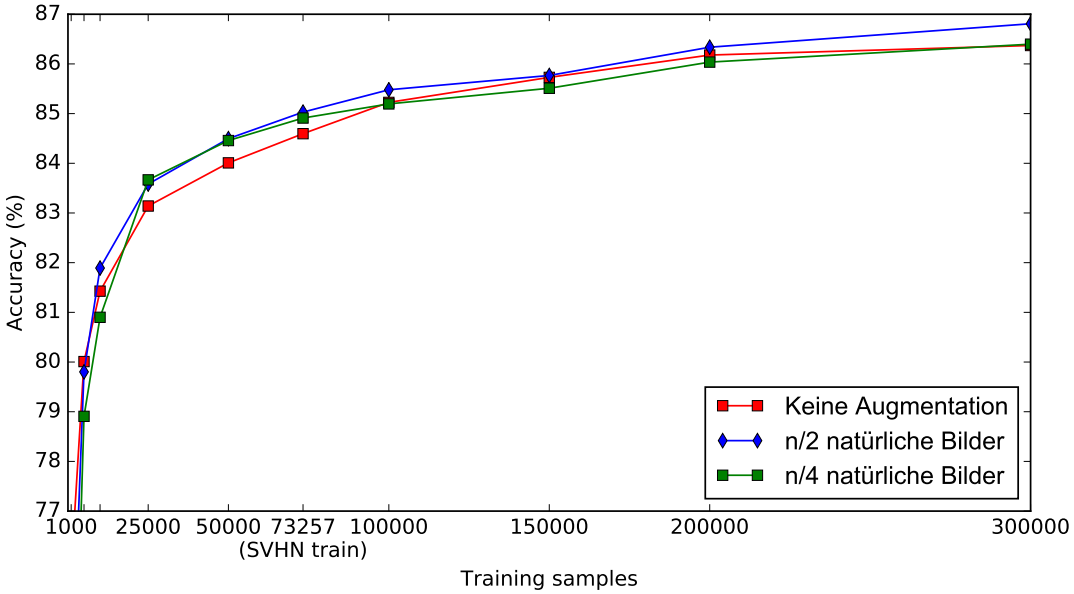


Abbildung 5.4.3: Erkennungsraten für verschiedene Zusammensetzungen der Trainingsmenge bei unterschiedlich großen Trainingsmengen.

Für alle drei Kompositionen der Trainingsmenge kann eine Vergrößerung der Trainingsmenge die Erkennungsrate steigern. Ab einer Menge von 50 000 Bildern erzielt die zur Hälfte aus augmentierten Bildern bestehende Trainingsmenge die besten Ergebnisse. Die höchste Erkennungsrate liegt für eine rein natürliche Trainingsmenge bei 86,37%, und für eine zur Hälfte aus augmentierten Bildern bestehende Menge bei 86,81%. Die besten Ergebnisse werden im Unterabschnitt 5.4.3 genauer betrachtet.

5.4.3 *Auswertung und Diskussion*

Im Folgenden sollen die Ergebnisse der Klassifikationsphase analysiert und diskutiert werden. Für alle drei Kompositionen der Trainingsmenge konnte eine Erhöhung der Gesamtzahl Bilder die Klassifikationsleistung verbessern. Es wurden Trainingsmengen mit n Bildern verglichen, die zu verschiedenen Anteilen aus augmentierten

Bildern bestehen. Die Ergebnisse zeigen, dass eine Augmentierung der Trainingsmenge die Klassifikationsleistung steigern kann. Die beste Klassifikationsleistung konnte mit einer Trainingsmenge erreicht werden, die aus 150 000 natürlichen und 150 000 augmentierten Bildern zusammengesetzt wurde. Zum Trainieren wurden die in Abschnitt 5.2 aufgeführten Einstellungen für die HOG-Deskriptoren und den Klassifikator genutzt. Die Konfusionsmatrix wird in Abbildung 5.4.4 gezeigt, und die Statistiken für Precision, Recall und den F1-Score in Tabelle 5.4.2 aufgeführt.

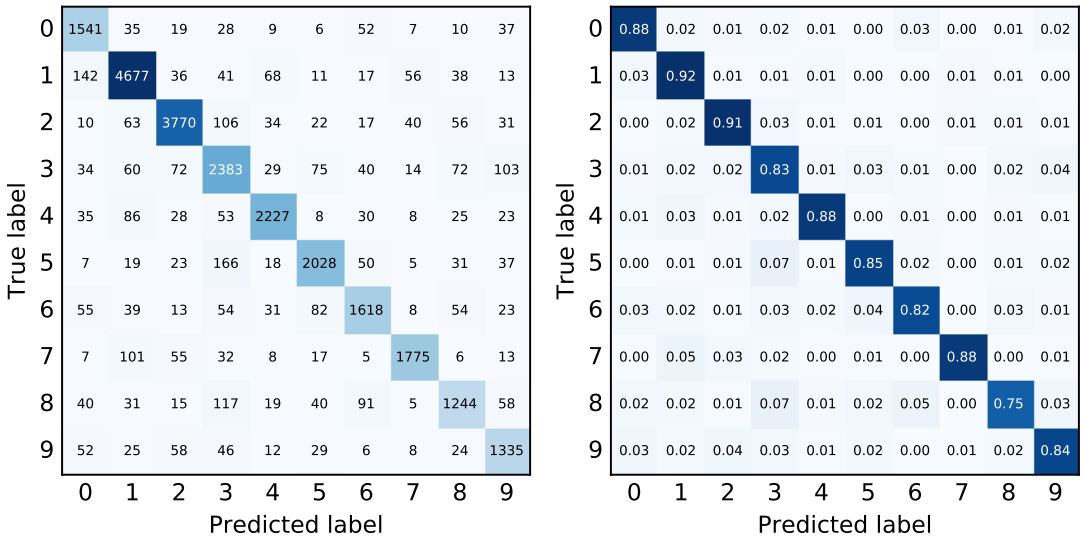


Abbildung 5.4.4: Die Konfusionsmatrizen für den mit der erweiterten Trainingsmenge trainierten Klassifikator. Links die absoluten Zahlen, rechts spaltenweise normiert.

Es ist erkennbar, dass zwischen den einzelnen Klassen Unterschiede in den Erkennungsraten bestehen. Die Konfusionsmatrix zeigt, dass zwischen der am besten erkannten Klasse, 1, und der am schlechtesten abschneidenden Klasse, 8, über 15% Unterschied in der Erkennungsrate bestehen. Die Klasse 8 zeigt einen besonders niedrigen Recall auf. Viele Objekte der Klasse 8 werden fälschlicherweise den Klassen 3 oder 6 zugeordnet. Die Klassen 0 und 3 haben einen auffällig niedrigen Precision-Wert: Viele Objekte der Klasse 0 werden als Objekte der Klassen 1, 8 oder 9 eingestuft, besonders viele Objekte der Klasse 3 zu der Klasse 5. Diese Verwechslungen lassen sich auf die visuelle Ähnlichkeit der Ziffern untereinander zurückführen. Die Ziffern für 3 und 8 sind z. B. in vielen Schriftarten visuell sehr ähnlich. Zusätzlich ist es wahrscheinlich, dass der Klassifikator durch die nicht ausbalancierten Klassen der Trai-

Klasse	Precision	Recall	F1-Score
0	80,14%	88,36%	84,05%
1	91,06%	91,72%	91,39%
2	92,20%	90,87%	91,53%
3	78,75%	82,69%	80,67%
4	90,71%	88,27%	89,47%
5	87,49%	85,07%	86,26%
6	84,01%	81,84%	82,91%
7	92,16%	87,91%	89,99%
8	79,74%	74,94%	77,27%
9	79,80%	83,70%	81,70%
⌀, ungew.	85,61%	85,54%	85,52%
⌀, gew.	86,92%	86,81%	86,83%

Tabelle 5.4.2: Precision, Recall und der F1-Score für jede Klasse. Wie in Tabelle 5.3.1 sind für jede Klasse die ungewichteten und gewichteten Mittelwerte angegeben.

ningsmenge (siehe Abbildung 4.1.1) für die Klassifikation die häufiger vorkommenden Klassen bevorzugt. Für die Klassen 1 und 2 ist die Erkennungsrate insgesamt am höchsten. Für häufig verwechselte Klassen werden einige falsch klassifizierte Beispielobjekte in Tabelle 5.4.3 gezeigt. Dies lässt sich durch die hohe Repräsentation von Einsen und Zweien sowohl in der Trainingsmenge wie auch der Testmenge (siehe Abbildung 4.1.1) zurückführen.

Die höchste erreichte Klassifikationsrate beträgt 86,81%. Damit konnte die in [NWC+11] für die HOG-Deskriptoren erreichte Erkennungsrate von 85,00% leicht übertroffen werden. Durch die gewählte Parameterkonfiguration und den Einsatz der Augmentationen war es möglich, mit einer bedeutend kleineren Trainingsmenge diese Ergebnisse zu erzielen: Während Netzer et. al etwa 500 000 Trainingsbilder verwenden [NWC+11], ist es möglich, mit nur 75 000 der natürlichen Trainingsbilder vergleichbare Ergebnisse zu erzielen.

Durch Verfahren, die auf neuronalen Netzen basieren ist es jedoch möglich, noch höhere Klassifikationsraten zu erreichen ([NWC+11], [SCL12]). Der Einsatz augmentierter Bilder könnte auch für *feature learning*-Verfahren die Ergebnisse leicht verbessern, und dabei mit deutlich weniger natürlichen, manuell annotierten Daten auskommen.

Klasse		Beispielbilder	Klasse		Beispielbilder
True	Pred.		True	Pred.	
5	3		**1**	**0**	
8	3		**2**	3	

Tabelle 5.4.3: Für die vier am häufigsten auftretenden Klassenverwechslungen werden jeweils drei Beispiele für falsch klassifizierte Bilder gezeigt. Das Klassenlabel unter True zeigt die tatsächliche Klasse des Objekts, das Label unter Pred. die vorhergesagte Klasse.

6 FAZIT

Die Erkennung von Hausnummern in natürlichen Szenen bietet viele praktische Anwendungsmöglichkeiten für den täglichen Gebrauch. Im Rahmen der Erstellung der Fotos für Dienste wie Google Street View werden automatisch eine große Menge von Fotos von Gebäudefassaden erstellt, aus denen automatisch Hausnummern extrahiert werden können ([NWC+11]). Durch die Erkennung der Hausnummernbilder lassen sich die erfassten Hausnummern ihren zugehörigen Gebäuden zuordnen, wodurch Karten und Navigationsdienste verbessert werden können (siehe Kapitel 1). Weiterhin ist eine vollautomatische Erkennung der Hausnummern sehr nützlich für die fortschreitende Automatisierung in der Technik, sowie zur Unterstützung blinder und sehbehinderter Menschen.
Um diese Möglichkeiten zu realisieren ist es nötig, beliebige Hausnummern aus Fotos vollautomatisch zu lokalisieren und zu klassifizieren. Dieses Problem besteht aus zwei verschiedenen Teilproblemen; der Lokalisierung von Hausnummern innerhalb von Fotos, sowie das Erkennen der lokalisierten Hausnummern. Das Erkennen der Hausnummern kann auf Ziffernebene [NWC+11, SCL12] oder auf der Ebene ganzer Zahlen [GBI+13] vollzogen werden. Für das Lokalisieren der Hausnummern wurde in [NWC+11] ein auf einem gleitenden Fenster basierender Algorithmus verwendet.
In dieser Bachelorarbeit wurde das Klassifizieren der Hausnummern auf der Ebene einzelner Ziffern betrachtet. Auf Grundlage der Arbeit von [NWC+11] wurde in dieser Arbeit ein Klassifikator erstellt, der in der Lage ist, Bilder einzelner Ziffern der Hausnummern einzuordnen. Der Klassifikator wurde mit einer Menge aus Trainingsbildern vorbereitet, aus welchen die Histogram of Oriented Gradients Merkmalsrepräsentation [DT05] berechnet wurde. Als potenzielle Klassifikatoren kamen der KNN-Klassifikator und die lineare Support Vector Machine in Betracht. Die Support Vector Machine konnte im Vergleich zum KNN-Klassifikator bessere Ergebnisse erreichen, und wurde somit als Klassifikator ausgesucht. Es wurden für die HOG-Deskriptoren sowie die Klassifikatoren mehrere Hyperparameter bestimmt und evaluiert. Als Trainingsmenge wurde der SVHN-Datensatz verwendet, welcher in [NWC+11] vorgestellt wurde. Der Klassifikator wurde mit 32×32 Pixel großen Graustufenbildern einzelner Ziffern trainiert, aus welchen ihre jeweiligen HOG-Deskriptoren berechnet wurden.
Zusätzlich zu den natürlichen Bildern des SVHN-Datensatzes wurde der Effekt aug-

mentierter Bilder untersucht. Hierzu wurden Bilder aus dem SVHN-Datensatz dupliziert, maschinell verändert, und der Trainingsmenge wieder beigefügt. Die erfolgreichsten Augmentationsverfahren wurden in Abschnitt 4.3 beschrieben. Als besonders effektiv haben sich affine Transformationen, wie Skalierungen, Rotationen, Scherungen und Verschiebungen erwiesen, sowie Kontrastanpassungen und der Einsatz von Unschärfefiltern. Es konnte mit 300 000 Trainingsbildern und der in Kapitel 5 ermittelten Parameterkombination eine Erkennungsrate von 86,81% erreicht werden. Bedingt durch visuelle Ähnlichkeiten hat sich gezeigt, dass bestimmte Ziffern häufig miteinander verwechselt werden. Beispiele hierfür sind die Ziffern 3 und 8, die in vielen Schriftarten sehr ähnlich dargestellt werden. Eine mögliche Erweiterung der Arbeit ist es, nach Wegen zu suchen, die Erkennungsrate zwischen einzelnen Klassen auszugleichen, sodass solche Verwechslungen seltener vorkommen. Durch die in dieser Arbeit ermittelten Parameter konnte die in [NWC+11] mit den HOG-Deskriptoren und der linearen SVM erreichte Erkennungsrate von 85,00% übertroffen werden. Es hat sich gezeigt, das nur 75 000 natürliche Bilder notwendig sind, um über 86% Erkennungsrate zu erreichen. Der Rest der Trainingsmenge wurde vollautomatisch durch Augmentationen erzeugt. Hierdurch lässt sich der manuelle Annotationsaufwand stark senken. Dies kann ggf. für vergleichbare Probleme mit wenigen verfügbaren Trainingsdaten von großer Bedeutung sein: Datensätze mit einer geringen Menge von Trainingsdaten können durch Augmentationen erweitert werden, um die Klassifikationsleistung für solche Probleme zu verbessern.
Das Verwenden einer HOG-Merkmalsrepräsentation hängt jedoch in jedem Fall hinter den vorrangingen in [NWC+11] und [SCL12] beschriebenen Methoden zurück. Eine mögliche Erweiterung zu dieser Arbeit ist daher, für weitere, performantere Verfahren die in dieser Arbeit vorgestellten Augmentationsverfahren für die Trainingsmenge zu verwenden.
Eine weitere mögliche Erweiterung dieser Arbeit ist es, die natürlichen Trainingsdaten gänzlich auszulassen, und die Ziffernbilder komplett synthetisch zu generieren. Andere Arbeiten im Bereich der Szenentexterkennung konnten, durch komplett synthetisch generierte Trainingsdaten, bereits sehr vielversprechende Ergebnisse erzielen ([JSVZ14]). Spezifisch für Hausnummern ist der Einsatz synthetischer Trainingsdaten jedoch noch unerforscht.

LITERATURVERZEICHNIS

[Bai93] Baird, Henry S.: Document image defect models and their uses. In: *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on* IEEE, 1993, S. 62–67

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer New York, 2006

[BMK14] Ba, Jimmy ; Mnih, Volodymyr ; Kavukcuoglu, Koray: Multiple object recognition with visual attention. In: *arXiv preprint arXiv:1412.7755* (2014)

[BT90] BT, RECOMMENDATION ITU-R: Basic parameter values for the HDTV standard for the studio and for international programme exchange. (1990)

[CBHK02] Chawla, Nitesh V. ; Bowyer, Kevin W. ; Hall, Lawrence O. ; Kegelmeyer, W P.: SMOTE: synthetic minority over-sampling technique. In: *Journal of artificial intelligence research* 16 (2002), S. 321–357

[CBV09] Campos, Teo de ; Babu, Bodla R. ; Varma, Manik: Character recognition in natural images. (2009)

[CMGS10] Cireşan, Dan C. ; Meier, Ueli ; Gambardella, Luca M. ; Schmidhuber, Jürgen: Deep, big, simple neural nets for handwritten digit recognition. In: *Neural computation* 22 (2010), Nr. 12, S. 3207–3220

[DHS01] Duda, Richard O. ; Hart, Peter E. ; Stork, David G.: *Pattern Classification, 2nd Ed*. Wiley, New York, 2001

[DT05] Dalal, Navneet ; Triggs, Bill: Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* Bd. 1 IEEE, 2005, S. 886–893

[DT17] DeVries, Terrance ; Taylor, Graham W.: Dataset Augmentation in Feature Space. In: *arXiv preprint arXiv:1702.05538* (2017)

[FCA+09] FROME, Andrea ; CHEUNG, German ; ABDULKADER, Ahmad ; ZENNARO, Marco ; WU, Bo ; BISSACCO, Alessandro ; ADAM, Hartwig ; NEVEN, Hartmut ; VINCENT, Luc: Large-scale privacy protection in google street view. In: *Computer Vision, 2009 IEEE 12th International Conference on* IEEE, 2009, S. 2373–2380

[GBI+13] GOODFELLOW, Ian J. ; BULATOV, Yaroslav ; IBARZ, Julian ; ARNOUD, Sacha ; SHET, Vinay: Multi-digit number recognition from street view imagery using deep convolutional neural networks. In: *arXiv preprint arXiv:1312.6082* (2013)

[Gol91] GOLDBERG, David: What every computer scientist should know about floating-point arithmetic. In: *ACM Computing Surveys (CSUR)* 23 (1991), Nr. 1, S. 5–48

[GW02] GONZALEZ, Rafael C. ; WOODS, Richard E.: *Digital image processing, 2nd Ed*. Prentice hall New Jersey, 2002

[GY01] GAO, Jiang ; YANG, Jie: An adaptive algorithm for text detection from natural scenes. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* Bd. 2 IEEE, 2001, S. II–II

[JCWJ07] JIANG, Liangxiao ; CAI, Zhihua ; WANG, Dianhong ; JIANG, Siwei: Survey of improving k-nearest-neighbor for classification. In: *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on* Bd. 1 IEEE, 2007, S. 679–683

[JSVZ14] JADERBERG, Max ; SIMONYAN, Karen ; VEDALDI, Andrea ; ZISSERMAN, Andrew: Synthetic data and artificial neural networks for natural scene text recognition. In: *arXiv preprint arXiv:1406.2227* (2014)

[JVZ14] JADERBERG, Max ; VEDALDI, Andrea ; ZISSERMAN, Andrew: Deep features for text spotting. In: *European conference on computer vision* Springer, 2014, S. 512–528

[KWTM97] KIMURA, Fumitaka ; WAKABAYASHI, Tetsushi ; TSURUOKA, Shinji ; MIYAKE, Yasuji: Improvement of handwritten Japanese character recognition using weighted direction code histogram. In: *Pattern recognition* 30 (1997), Nr. 8, S. 1329–1337

[Mur12] MURPHY, Kevin P.: Machine learning: a probabilistic perspective. (2012)

[NFHS12] NISCHWITZ, Alfred ; FISCHER, Max ; HABERÄCKER, Peter ; SOCHER, Gudrun: *Computergrafik und Bildverarbeitung, Band II: Bildverarbeitung, 3. Auflage*. Vieweg+Teubner Verlag, 2012

[NWC+11] NETZER, Yuval ; WANG, Tao ; COATES, Adam ; BISSACCO, Alessandro ; WU, Bo ; NG, Andrew Y.: Reading digits in natural images with unsupervised feature learning. In: *NIPS workshop on deep learning and unsupervised feature learning* Bd. 2011, 2011, S. 5

[PCN10] POSNER, Ingmar ; CORKE, Peter ; NEWMAN, Paul: Using text-spotting to query the world. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* IEEE, 2010, S. 3181–3186

[PLH10] PAN, Yi-Feng ; LIU, Cheng-Lin ; HOU, Xinwen: Fast scene text localization by learning-based filtering and verification. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on* IEEE, 2010, S. 2269–2272

[Pri15] PRIESE, Lutz: *Computer Vision*. Springer Vieweg, 2015

[Ras15] RASCHKA, Sebastian: *Python Machine Learning*. Birmingham, UK : Packt Publishing, 2015. – ISBN 1783555130

[SCL12] SERMANET, Pierre ; CHINTALA, Soumith ; LECUN, Yann: Convolutional neural networks applied to house numbers digit classification. In: *Pattern Recognition (ICPR), 2012 21st International Conference on* IEEE, 2012, S. 3288–3291

[SSP+03] SIMARD, Patrice Y. ; STEINKRAUS, David ; PLATT, John C. u. a.: Best practices for convolutional neural networks applied to visual document analysis. In: *ICDAR* Bd. 3, 2003, S. 958–962

[Sze10] SZELISKI, Richard: *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010

[TK08] THEODORIDIS, Sergios ; KOUTROUMBAS, Konstantinos: *Pattern Recognition, Fourth Edition*. 4th. Academic Press, 2008. – ISBN 1597492728, 9781597492720

[TN17] TAYLOR, Luke ; NITSCHKE, Geoff: Improving Deep Learning using Generic Data Augmentation. In: *arXiv preprint arXiv:1708.06020* (2017)

[WB10] Wang, Kai ; Belongie, Serge: Word spotting in the wild. In: *European Conference on Computer Vision* Springer, 2010, S. 591–604

[WGSM16] Wong, Sebastien C. ; Gatt, Adam ; Stamatescu, Victor ; McDonnell, Mark D.: Understanding data augmentation for classification: when to warp? In: *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on* IEEE, 2016, S. 1–6

[YD15] Ye, Qixiang ; Doermann, David: Text detection and recognition in imagery: A survey. In: *IEEE transactions on pattern analysis and machine intelligence* 37 (2015), Nr. 7, S. 1480–1500

[YLW97] Yaeger, Larry S. ; Lyon, Richard F. ; Webb, Brandyn J.: Effective training of a neural network character classifier for word recognition. In: *Advances in neural information processing systems*, 1997, S. 807–816

[ZWWW15] Zhang, Yuqi ; Wang, Wei ; Wang, Liang ; Wang, Liuan: Scene text recognition with deeper convolutional neural networks. In: *Image Processing (ICIP), 2015 IEEE International Conference on* IEEE, 2015, S. 2384–2388

[ZYB16] Zhu, Yingying ; Yao, Cong ; Bai, Xiang: Scene text detection and recognition: Recent advances and future trends. In: *Frontiers of Computer Science* 10 (2016), Nr. 1, S. 19–36