TECHNISCHE UNIVERSITÄT WIEN

ACIN
AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Inkrementelles Lernen von Objektklassen mittels Neuronaler Netze

DIPLOMARBEIT

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs (Dipl.-Ing.)

unter der Leitung von

Ao. Univ.-Prof. Dr. techn. M. Vincze
Dipl.-Ing. T. Fäulhammer

eingereicht an der

Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungs- und Regelungstechnik

von
Martin Palkovits
Große Mohrengasse 20/4/20
1020 Wien
Österreich

Wien, im Mai 2017

Vision for Robotics Gruppe
A-1040 Wien, Gusshausstr. 27, Internet: http://www.acin.tuwien.ac.at

Vorwort

Diese Arbeit ist im Rahmen des Vision for Robotics(v4r) Projektes des Institutes für Automatisierungs- und Regelungstechnik entstanden. Im Rahmen dieser Arbeit war es mir möglich mein schon zuvor gewecktes Interesse an der visuellen Bildverarbeitung zu vertiefen. Ich fand interessant zu sehen wie Klassifizierungsalgorithmen verbessert und verändert werden können und welche Performance mit ihnen möglich ist. Es war für mich auch sehr spannend zu sehen wie der von außen scheinbar recht komplexe Vorgang der Klassifizierung elegant durch den Computer gelöst wurde.

Ich möchte mich bei Prof. Markus Vincze für die Möglichkeit der Durchführung sowie die Bereitstellung eines Arbeitsplatzes bedanken. Bei Dipl.-Ing. Thomas Fäulhammer möchte ich mich für die Bereitstellung des Themas sowie die Betreuung der Arbeit bedanken. Ebenfalls möchte mich bei Dr. techn. Michael Zillich für das Einspringen bei der Betreuung während der Abwesenheit von Herrn Fäulhammer bedanken. Weiters möchte mich bei meiner Familie bedanken, die mich während dieser Arbeit und im gesamten Studium immer unterstützt hat und mit Rat und Tat bereit gestanden ist.

Wien, im Mai 2017

Abstract

Classifying objects is a very complex procedure due to the big variety of objects and their geometric and visual characteristics. Training an object classifier that is able to distinguish a large number of different object classes typically requires a tremendous amount of computational effort and training data. To partially overcome these problems, an iterative training method is proposed that improves recognition of frequently seen objects. Using a pre-trained convolutional neural network, backpropagation and stochastic gradient descent is used to update the network on every image of the dataset, which is not classified with an high enough confidence. To avoid biasing the classifier towards reappearing objects, we propose *self-check*. This method evaluates the classifier at every iterative training process on a subset of the initial training set to check if all classes are recognised correctly. Additionally, we use different data augmentation methods in different parts of the classification and training processes and show that if the images are augmented before the classification, the number of correct classified images decreases. We evaluated our method on the cat-10 dataset of the 3DNet database and show that by using the proposed iterative classification method the number of correctly classified objects can be improved by $20 - 30\%$.

Kurzzusammenfassung

Aufgrund der großen Varietät an Objekten und deren geometrischen sowie visuellen Eigenschaften, ist die exakte Klassifizierung von Objekten ein sehr komplexer Vorgang. Um einen Klassifizierungsalgorithmus so zu trainieren, sodass er zwischen einer Vielzahl von unterschiedlichen Objektklassen unterscheiden kann ist ein sehr großer Rechenaufwand sowie eine enorme Menge an Trainingsdaten erforderlich. Um diese Probleme teilweise zu umgehen, wurde eine iterative Trainingsmethode verwendet, um das Wiedererkennen häufig gesehener Objektklassen zu verbessern. Ausgehend von einem schon trainierten convolutional neural network, wurde mittels backpropagation und stochastic gradient das Netz bei jenen Bildern upgedatet, bei welchen die Konfidenz der Klassifizierung zu gering war. Um einen mögliches biasen des Klassifizierers zu vermeiden wurde eine neue Methode *self-check* angewandt. Diese Methode evaluiert den Klassifizierer bei jedem iterativen Trainingsprozess auf einem Subset des anfänglichen Trainingssets, um so zu überprüfen ob alle Klassen noch korrekt wiedererkannt werden. Zusätzlich haben wir auch unterschiedliche Bildaugmentierungsmethoden zu unterschiedlichen Zeitpunkten während der Klassifizierung beziehungsweise des iterativen Trainingsprozesses angewandt und konnten zeigen, dass sofern die Bilder vor der Klassifizierung augmentiert werden, die Anzahl an korrekt Klassifizierten Bildern abnimmt. Wir haben unsere verwendete Methode auf der cat-10 Datenbank des 3DNet Datensatzes evaluiert und zeigen das durch Anwendung iterativen Trainings die Anzahl an korrekt erkannten Objekte um $20 - 30\%$ gesteigert werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In den letzten Jahren sind Roboter ein immer größerer Bestandteil unseres Lebens und unserer Gesellschaft in sehr unterschiedlichen Bereichen geworden. Zum Beispiel in der Altenpflege(HOBBIT), Heimunterhaltung (AIBO), Security (SAM) oder als Haushaltshilfen(ROOMBA). Damit eine möglichst effektive Unterstützung durch Roboter möglich ist, müssen sie sich in unserer Welt zurecht finden und mit Dingen interagieren können. Bei diesen beiden Vorgängen spielt das Erkennen beziehungsweise Wiedererkennen von Objekten oder Gegenständen eine große Rolle. Das Erkennen von einzelnen Objekten ist aufgrund der hohen Varianz selbst ähnlicher Objekte, also kleiner Unterschiede, welche den Verwendungszweck des Objekts nicht verändern, z.B. Unterschiede in der Farbe, nicht immer sinnvoll(Abbildung 1.3). Daher ist eine Klassifizierung von Objekten notwendig.

Die Performance des Klassifizierers hängt aufgrund der unterschiedlichen Charakteristiken von Objekten stark vom gewählten Szenario und Kriterium ab. Man unterscheidet grundsätzlich zwischen dem Erkennen von Objektinstanzen und der Unterscheidung von Objektklassen. Beim Erkennen von Objektinstanzen ist man an einer bestimmten Instanz eines Objektes interessiert (z. B. roter Ferrari Typ 488 BJ 2005). Beim Unterscheiden von Objektklassen, womit wir uns in dieser Arbeit beschäftigen, werden die Objekte in Kategorien, beispielsweise Auto, Boot, Motorrad usw. eingeteilt.

Um ein Objekt zu klassifizieren ist es zunächst notwendig markante und unterscheidbare Merkmale der Objekte, welche in einem Deskriptor(x) dargestellt werden, zu beschreiben. Mit diesen Merkmalen wird dann ein Klassifizierungsproblem erstellt und nach einem gewissen Kriterium optimiert. Dieser Vorgang wird auch Trainingsprozess genannt. Bei diesem Prozess wird der Klassifizierer(f) auf den Trainingsdaten optimiert. Der prinzipielle Ablauf eines Klassifizierungsvorganges ist in Abbildung 1.1 dargestellt.

Bei den Trainingsdaten handelt es sich in dieser Arbeit um Bilder oder 3D-Pointclouds von Objekten, welche Klassen repräsentieren, die unterschieden werden sollen. Diese werden mittels Deskriptoren(x) kodiert und mit den zugehörigen Trainingslabels, oder auch Ground-Truth Labels($\hat{y}$), kann der Klassifizierungs-Algorithmus trainiert werden.

Die Menge und Art der Trainingsdaten spielt eine wichtige Rolle für die Güte des entstehenden Klassifizierungsalgorithmus. Je größer und variantenreicher

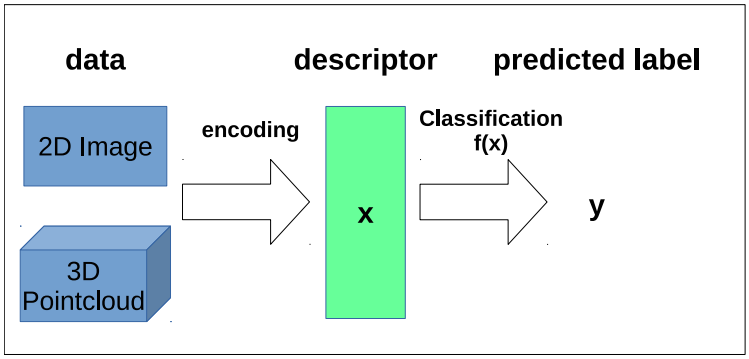
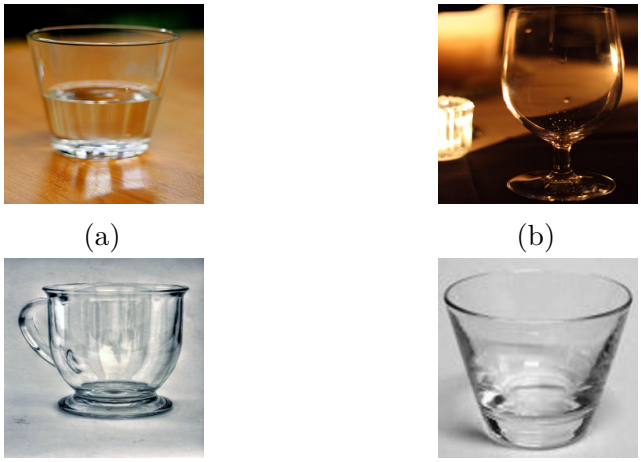


Abbildung 1.1: Schematische Darstellung des Ablaufs von Objekt Klassifizierung

diese Datenbank ist, umso mehr Objektklassen als auch -instanzen werden vom Algorithmus richtig klassifiziert werden. Eine große Datenbank stellt allerdings nicht immer sicher, dass alle Objekte richtig klassifiziert werden. Die sogenannte inter-class-similarity oder Ähnlichkeiten von Objekten unterschiedlicher Klassen kann ein Problem darstellen, ein Beispiel ist in Abbildung 1.2 gegeben. Weiters kann intra-class-variance oder Unterschiede zwischen Objekten derselben Klasse den Klassifizierer auch verunsichern, ein Beispiel ist in Abbildung 1.3 gegeben. Jede Trainingsdatenbank sollte diese beiden Effekte berücksichtigen, da so eher die Parameter des Trainingsprozesses den entstehenden Klassifizierer beeinflussen.



(a) (b)

(c) (d)

Abbildung 1.2: Beispiel für inter-class-similarity in Form von Objekten aus Glas welche unterschiedlichen Klassen zugeordnet werden sollen: Glas(a,b), Tasse(c) und Schüssel(d)

Abbildung 1.3: Beispiel für intra-class-variance in Form von Autos welche der selben Klasse angehören, aber über sehr unterschiedliche Formen und Farben verfügen

1.1 Problemstellung

Im Rahmen dieser Arbeit wird von einem Klassifizierungsalgorithmus f welcher durch den Parametersatz $\mathbf{w} \in \mathbb{R}^o$, wobei o die Anzahl an Parametern des Algorithmus angibt, beschrieben wird. Die zu klassifizierenden Daten $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{d+1}$, wobei d die Dimension des Deskriptors angibt, werden von dem Klassifizierer auf ein Label $y \in \mathcal{Y} = \{class_0, class_1, ..., class_m\}$, wobei m die Anzahl an Klassen angibt, abgebildet.

$$f : \mathcal{X} \Rightarrow \mathcal{Y} \tag{1.1}$$

Ziel des *Trainings* ist es den Trainingsfehler $E_{\text{train}}(\mathbf{w})$ zu minimieren. Dieser Fehler besteht aus der Summe der Differenzen der klassifizierten Trainingsdaten $x_1, x_2, ..., x_n$ und der jeweiligen korrekt annotierten Labels $\hat{y}_1, \hat{y}_2, ..., \hat{y}_n$, wobei n die Anzahl an Trainingsdaten angibt. Der Parametersatz, welcher im Rahmen des ersten Trainingsprozesses ermittelt wurde wird als $\mathbf{w}_0$ bezeichnet.

$$\mathbf{w}_0 = \underset{\mathbf{w}}{\operatorname{argmin}} E_{\text{train}}(\mathbf{w}) = \sum_{i=1}^{n} (\hat{y}_i - f(x_i, \mathbf{w}))^2 \tag{1.2}$$

Ziel dieser Arbeit ist die Entwicklung effizienter Methoden zur Minimierung des Testfehlers $E_{\text{test}}(\mathbf{w})$. Hierbei handelt es ich ähnlich dem Trainingsfehler um die Summe der Differenzen der klassifizierten Testdaten $x_1^t, x_2^t, ..., x_m^t \in \mathcal{X}_{\text{test}} \subset \mathcal{X}$ und korrekten Labels $\hat{y}_1^t, \hat{y}_2^t, ..., \hat{y}_m^t$, wobei m die Anzahl an Trainingsdaten angibt.

$$E_{\text{test}}(\mathbf{w}) = \sum_{X_{\text{test}}} (\hat{y}_i^t - f(x_i^t, \mathbf{w}))^2 \tag{1.3}$$

Die Reduktion dieses Fehlers soll durch zusätzliches Training auf einem weiteren Trainingsset $x_1', x_2', ..., x_r' \in \mathcal{X}_{\text{train'}} \subset \mathcal{X}$, wobei $\mathcal{X}_{\text{test}} \cap \mathcal{X}_{\text{train'}} = \emptyset$ und

den jeweiligen vom Benutzer korrekt annotierten Labels $\hat{y}'_1, \hat{y}'_2, ..., \hat{y}'_r$ erreicht werden. Der Parametersatz, welcher im Rahmen des zusätzlichen Trainings ermittelt wurde wird als $\mathbf{w}^*$ bezeichnet.

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{X_{\text{train'}}} (\hat{y}'_i - f(x'_i, \mathbf{w}))^2 \tag{1.4}$$

$$E_{\text{test}}(\mathbf{w}^*) < E_{\text{test}}(\mathbf{w}_0) \tag{1.5}$$

Bei diesen Trainingsprozessen kann der Klassifizierer gebiast werden. Bei einem sogenannten Bias-Fehler handelt sich um ein Verzerren des Klassifizierers hin zu einer Klasse, sodass auch Objekte anderer Klassen als Objekte dieser überrepräsentierten Klasse erkannt werden.

Grundsätzlich kann für einen solchen Trainingsprozess zwischen unterschiedlichen Herangehensweisen unterschieden werden:

- Zum einen kann die Klasse oder Instanz in die Trainingsdatenbank aufgenommen werden und es wird ein komplett neuer Klassifizierer trainiert. Dieser Vorgang wird als *batch-learning* bezeichnet.
- Zum anderen kann sogenanntes *online-learning* durchgeführt werden, wobei ein vorher trainierter Klassifizierer durch neue annotierte Daten X_{new} adaptiert wird.

Beim *Batch-learning* besteht der Vorteil darin, dass der Klassifizierungsalgorithmus immer auf allen bekannten Daten zeitgleich trainiert wird. Ein Bias-Fehler ist also am ehesten durch eine unzureichende Trainingsdatenbank, z. B. durch Überrepräsentation einer Klasse möglich. Alerdings werden für diesen Vorgang auch immer alle Daten benötigt, was in einem sehr großen Speicherbedarf resultiert. Der Hauptgrund, warum *batch-learning* nicht zur Anwendung bei iterativem oder inkrementellem Lernen kommt, ist der große Zeitaufwand, der hierfür benötigt wird. Dieser kann im Fall von AlexNet[3], welches auf den ImageNet[4] Daten trainiert wird, mehrere Tage bis Wochen betragen und ist daher nicht für zeitkritische Anwendungen geeignet.

Beim *Online-learning* ist der Vorteil, dass theoretisch keinerlei Daten der Trainingsdatenbank benötigt werden, da lediglich auf Daten trainiert wird, welche z. B. nicht korrekt klassifiziert wurden. Dies resultiert auch in einem deutlich geringeren Rechenaufwand für das erneute Trainieren. Allerdings kann es hier neben dem durch eine ungleichmäßig aufgeteilte Datenbank auch zu einem Bias-Fehler kommen, falls ein Objekt sehr oft hintereinander trainiert wird. Daher werden augmentierte Trainingsdaten mitgeführt, um auf einen solchen Bias-Fehler testen zu können.

Das Ziel dieser Diplomarbeit ist die Entwicklung eines Algorithmus, welcher durch online-learning verbessert wird.

1.2 Related Work

Objektklassifizierung ist seit vielen Jahren ein wichtiger Bestandteil von Computer Vision Systemen. Prominente Methoden können unter anderem in:

- k-Nearest Neighbor(k-NN) search[5]
- Support Vector Machines(SVM)[6]
- Random Forest[7]
- Neural Networks(NN)[3]

eingeteilt werden.

In [8] wurde die Klassifizierung durch k-NN search, SVM und Random Forest von Gesichtsausdrücken verglichen. Es wurde gezeigt, dass mit je mehr Daten im Rahmen des Trainingsprozesses verwendet wurden, sich die Performance von k-NN search und Random Forest gegenüber der von SVM deutlich verbesserte. Diese Verbesserung ist auf eine nicht klare Trennbarkeit der Klassen zurückgeführt worden. Da in der von uns verwendeten Datenbank eine klare Trennung möglich ist, war für uns die Klassifizierung mittels k-NN search oder Random Forest nicht weiter interessant.

In [9] wurden unterschiedliche Arten von Klassifizierungsalgorithmen für die Anwendung auf RGB-D verglichen. Es wurden zwei lokale Deskriptoren(SIFT[10] und SHOT[11]) und zwei globale(ESF[12] und ein Alogrithmus basierend auf AlexNet[3]) verwendet. Die jeweiligen Deskriptoren wurden einzeln entweder durch kNN(SIFT, SHOT, ESF) oder durch eine SVM(AlexNet) klassifiziert. Anschließend wurden die Ergebnisse der lokalen und globalen Deskriptoren zusammengeführt und validiert. Es konnte gezeigt werden, dass der Zusammenschluss der einzelnen Ergebnisse die Klassifizierung verbessert. Allerdings ist für uns aufgrund des hier verwendeten iterativen Trainings nicht zielführend mehrere Klassifizierer zu verwenden, da als Bedingung ob iteratives Training angewandt werden soll, die Konfidenz der Klassifizierung als entscheidender Parameter herangezogen wird. So wird die Entscheidung, welcher Klassifizierer iterativ trainiert werden soll erschwert.

Für die Computer Vision Community haben vorallem convolutional neural networks(CNN) in den letzen Jahren immer größere Bedeutung erlangt. Grund dafür sind unter anderem die von ihnen durchgeführten Faltungsoperationen, welche die Anzahl an benötigten Parametern reduziert und sie so sehr gut zur Verarbeitung von großen Datensätzen anwendbar machen. In unserer Arbeit befassen wir uns ebenfalls mit den Eigenschaften von CNNs - speziell in Bezug auf inkrementelles Lernen. Der Aufbau sowie die Funktionsweise von CNNs wird in Kapitel 2.1 näher beschrieben.

1.2.1 Neural Networks

In den letzten Jahren wurden vermehrt Neural Networks(NN) als Klassifizierungsalgorithmen herangezogen[3], [9], [13]–[15]. Diese eignen sich Aufgrund ihres Aufbaus, welcher in Kapitel 2.1 näher erläutert wird, und der damit einhergehenden Eigenschaften sehr gut zur Verarbeitung von großen Datensätzen. In [13] wurde ein Klassifizierer für Bilder, welche Tiefen-Daten enthalten entwickelt. Dieser basiert auf einem NN, wobei die Trainingsdaten mittels Rough-Sets auf jenen Daten, welche die meiste Information beinhalten, optimiert. Anschließend wurde dieser Klassifizierer mit anderen State-of-the-Art Klassifizierern verglichen, z. B. Support Vector Machine(SVM) und NN welche auf allen Daten trainiert wurde. Die Ergebnisse zeigen auf, dass neben einer Verkürzung der Trainingszeit auch eine Verbesserung in der Anzahl an korrekt klassifizierten Bildern feststellbar ist. Da bei dem von uns verwendeten Klassifizierer jedoch kein komplettes Netz, sondern lediglich ein Teil erneut trainiert wurde auf eine Optimierung der Daten durch Rough-Sets verzichtet. In [14] wurden NN verwendet um Teile von Satellitenbilder unterschiedlichen Klassen zuzuordnen. Hier wurden zwei unterschiedliche Methoden vorgestellt um die Klassifizierung zu verbessern. Zum einen wurde vorab verfügbare Information über die Trainingsbilder im Rahmen der Klassifizierung berücksichtigt, zum anderen wurde, da die zu klassifizierenden Teile in diesem Fall nicht immer exakt einer Klasse zugeordnet werden können, fuzzy-Mittelwerte bzw. -Varianzen der Klassen errechnet und in den Trainingsprozess mit einbezogen. Durch diese Maßnahmen konnte die Anzahl an korrekten Klassifizierungen deutlich gesteigert werden. Bei dem von uns betrachteten Fall sind die zu klassifizierenden Bilder allerdings eindeutig klassifizierbar, deshalb war keine Anwendung von Vorabinformation oder Fuzzylogik notwendig. In [15] wurden zwei CNNs(AlexNet und GoogLeNet[16]) verwendet um verschiedene Arten von medizinischen Bildern zu klassifizieren. Hier wurden die Netze zum einem verwendet um die Daten zu klassifizieren und zum anderen um Deskriptoren zu errechnen, welche anschließend in unterschiedlichen Kombinationen von einer SVM klassifiziert wurden. Die Ergebnisse dieser Arbeit zeigen, dass die Klassifizierung durch die Netze nicht deutlich weniger korrekte Ergebnisse liefert als die Klassifizierung der Deskriptoren durch die SVM. Weiters wurde in dieser Arbeit auch ermittelt, dass einer der Unterschiede der beiden verwendeten Netze in der Generalisierungsfähigkeit liegt. Da von uns die Generalisierungsfähigkeit getestet werden soll, bestätigt dies die Wahl von AlexNet als Klassifizierer. In [17] wurde die Performance unterschiedliche Klassifizierungsmethoden(u. a. SVM und NN) im Rahmen der Gasbläschen-Detektion für radiometrische Untersuchungen verglichen. Es wurden Bilder von Behältern, welche ein Gel, in welchem die Gasbläschen eingeschlossen waren erzeugt und sollten bezüglich der Anzahl an Bläschen

untersucht werden. Es konnte gezeigt werden, dass das NN der SVM sowie den anderen getesteten Methoden überlegen ist.

1.2.2 Online learning

Online learning oder inkrementelles Lernen ist ein grundlegendes Konzept, in dessen Rahmen versucht wird den Klassifizirer mit der Verwendung von einzelnen Daten durch Trainingsprozesse zu verbessern. Häufig wird Online learning in Zusammenhang mit Nearest Neighbour Algorithmen oder Support Vector Machines (SVM) verwendet, da sich diese Algorithmen aufgrund des Aufbaus sehr gut hierfür eignen[18]–[22].

Arbeiten, welche sich mit dem inkrementellen Lernen von neuronalen Netzen(NN) beschäftigen, weisen allerdings keinerlei Berücksichtigung eines möglichen Bias-Fehlers auf[23]–[30]. Dies war in den angeführten Arbeiten nicht notwendig, da in diesen Arbeiten gut durchmischte Testdaten verwendet wurden und so ein möglicher Bias gering gehalten wurde. Weiters wird sofern inkrementelles Lernen mit NN betrieben wird meist von Netzen ausgegangen, welche nur über drei Layer verfügen[24], [26], [28]–[30]. Von diesen Layern wird der mittlere häufig neben der Anpassung der Gewichte auch in der Größe verändert um so die Komplexität des Netzes verändern zu können[26], [28]–[30]. Da sich in dieser Arbeit allerdings um ein NN mit acht Layern handelt, welche sich in der Größe nicht verändern sollen sind die in den angeführten Methoden für diesen Klassifizierer nicht anwendbar. In der Arbeit von Polika, Upda et al.[23] wurde inkrementelles Lernen nicht durch ein Update von Gewichten erreicht, sondern durch das Hinzufügen von weiteren NN. Die verwendeten Netze wurden alle zusammen ausgewertet und durch Mittlung aller Ergebnisse wurde das Endresultat bestimmt. In [24] wurden drei online trainierte Netze verwendet um Bewegungsdaten eines Benutzers zu klassifizieren. Jeder Achse wurde ein Netz zugewiesen und iterativ trainiert. Hier wurde im Rahmen des iterativen Trainings ein Möglichkeit zur Selbstüberprüfung des Netzes vorgestellt. Dies ermöglichte eine Reduktion der Zeitdauer beim iterativen Training. Da beim hier entwickelten Klassifizierer lediglich einmal pro Bild iterativ trainiert werden sollte, war ein laufendes Überprüfen des Klassifizierers nicht notwendig Cui, Surpur et al. [25] präsentierten in ihrer Arbeit eine komplett neue Form von Neuronen, welche sich besonders für online Learning eignen und konnten mit einem Netz aus diesen eine bessere Klassifizierungsrate als bei herkömmlichen NN erreichen. Allerdings sind diese Neuronen auf laufendes Training bei allen klassifizierten Daten optimiert. Der im Rahmen dieser Arbeit entwickelte Klassifizierer, basierend auf AlexNet, muss allerdings nicht bei jedem klassifizierten Bild iterativ trainiert werden, wodurch AlexNet als Klassifizierer ausreicht. In [27] wurde eine neue Möglichkeit des iterativen Trainings vorgestellt, wel-

che durch die Verwendung von pseudo-inversen Matrizen die Berechnungszeit während des Trainingsprozesses reduziert.

1.3 Lösungsansatz

Der entwickelte Algorithmus basiert auf AlexNet[3], einem neuronalen Netzwerk, welches im Zuge der ImageNet Klassifizierungsaufgabe 2012[4] entstanden ist. Die ImageNet Datenbank[4] besteht aus mehreren Millionen Bildern, welchen mehr als 1000 verschiedene Objekte zugeordnet sind. AlexNet[3] klassifiziert ein 265x265 Pixel großes Bild in 1000 verschiedenen Klassen. Um diese Klassifizierung durchzuführen, werden in AlexNet die Daten des Bildes zunächst in fünf aufeinander folgenden convolutional Layern und anschließend in drei inner product Layern verarbeitet(siehe Abbildung 1.4). Die einzelnen Farbkanäle(rot,grün und blau) werden zunächst auch getrennt verarbeitet und erst später zusammen geführt. Die ersten sieben Layer berechnen mit zunehmender Layer-Zahl immer komplexere Features, zunächst werden beispielsweise Kanten und Ecken und gegen Ende Teile des Objekts wie Augen oder Räder erkannt, welche vom finalen Layer klassifiziert werden[32].

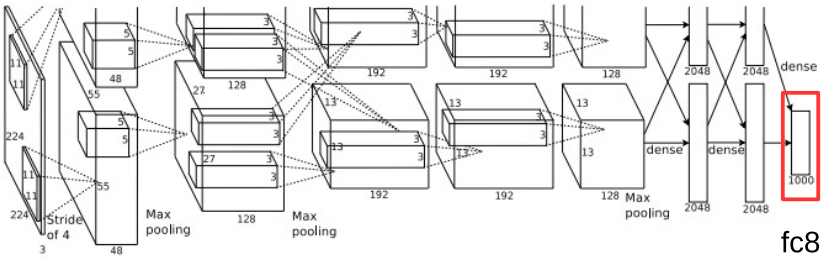


Abbildung 1.4: Darstellung der einzelnen Layer des AlexNet Netzwerks. Die einzelnen Zahlen repräsentieren die Dimensionen der einzelnen Layer.[3]

Anders als bei der ImageNet-Klassifizierungsaufgabe, wird in dieser Arbeit die Möglichkeit der Generalisierung des AlexNet Netzes an den Testdaten des 3DNet-Datensatzes[31] getestet. Im speziellen wurden die CAT10 Modelle welche 10 verschiedene Objekte zeigen verwendet. Daher war es Notwendig, den letzten Layer(fc8) von AlexNet so anzupassen, dass er nur 10 verschiedene Objektklassen unterscheiden kann. Für die ersten sieben Layer wurden die Gewichte, welche durch den Trainingsprozess auf der ImageNet-Datenbank[4] optimiert wurden, verwendet. Da diese Low-Level Features, z. B. Ecken oder Kanten enthalten[32], und durch ImageNet auf Millionen von Bildern trainiert wurden, ist eine signifikante Änderung der Gewichte durch weiteres Training

unwahrscheinlich. Der veränderte letzte Layer wurde auf einer Auswahl der ImageNet Bildern, welche die 10 Objekte der Testdatenbank zeigen, trainiert. Gupta, Girshick et. al[33] haben gezeigt, dass die letzten beiden Layer von AlexNet, wenn sie passend trainiert werden, gute Resultate bei Objektklassifizierung zeigen.

Als Testdatenbank wurde der Test-Teil der Cat10-Datenbank des 3D-Net Datensatzes[31] verwendet. Diese Datenbank zeigt über 1600 Pointclouds von 10 verschiedenen Objekten, welche einzeln auf einem Tisch dargestellt werden.

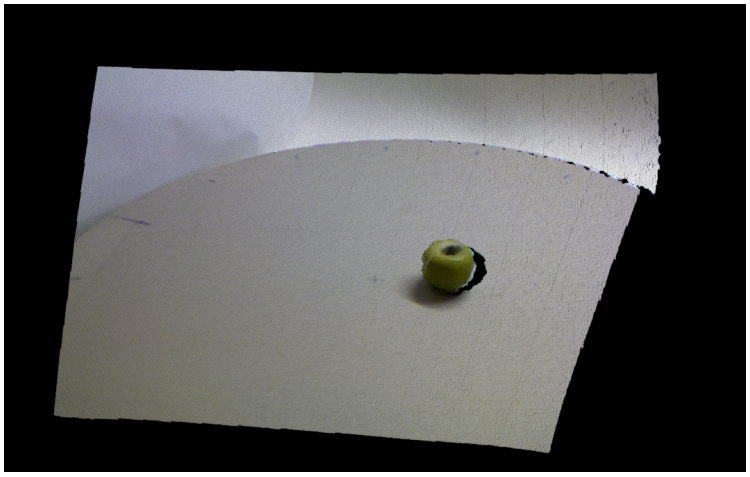


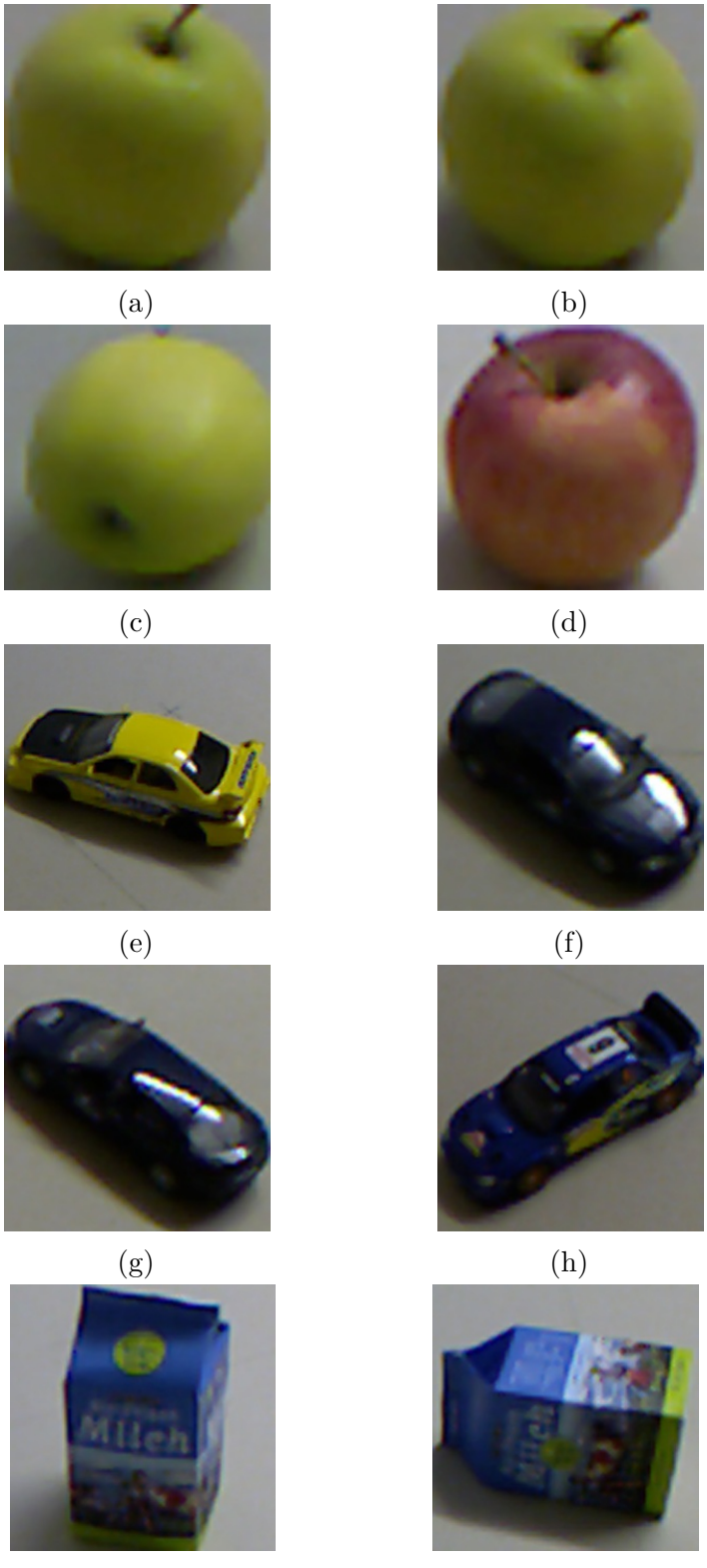
Abbildung 1.5: Beispiel einer Pointcloud der Testdatenbank

Die 10 Klassen sind folgende, wobei die Zahl in Klammern die Anzahl an Pointclouds des jeweiligen Objekts angibt:

1. Apfel (128)
2. Banane (73)
3. Flasche (324)
4. Schüssel (68)
5. Auto (154)
6. Donut (50)
7. Hammer (177)
8. Tasse (372)
9. Tetra Pack (172)

10. Toiletten Papier (94)

Diese Datenbank ist zur Evaluierung von Klassifizieren, welche Objekte über Tiefendaten klassifizieren und nicht Farbdaten, wie der Algorithmus in dieser Arbeit, entstanden. Daher unterscheiden sich die unterschiedlichen Objektklassen mehr in der Form als in der Farbe. Dieser Umstand macht die Datenbank interessant für Untersuchungen, wie sie in dieser Arbeit durchgeführt wurden. Abbildung 1.6 zeigt verschiedene Beispiele der Datenbank. Es sind eindeutig unterschiedliche Objekte aus derselben Klasse zu erkennen, welche auch aus unterschiedlichen Winkeln dargestellt sind. Die Umgebung der Objekte ist eher schlicht, was ein Segmentieren des Objekts erleichtert. Weiters zeigt jede Pointcloud nur ein Objekt mit Umgebung. Der im Rahmen dieser Arbeit entwickelte Algorithmus benötigt 256x256 Pixel große Bilder der zu klassifizierenden Objekte. Um dies zu erreichen, wurde zunächst mittels planarer Ebenensuche und RANSAC die Punkte des Tisches in der Pointcloud ermittelt. Aus den Punkten oberhalb dieser Ebene wurden Cluster gebildet und der Cluster mit dem geringsten Abstand zur Kamera, Ursprung der Pointcloud, als Objekt ausgewählt. Aus den Bildpunkten des selektierten Clusters werden 3 Bilder erstellt, eines zeigt die Punkte mit schwarzem(Abbildung 1.7a), eines mit weißem Hintergrund(Abbildung 1.7b). Beim dritten Bild(Abbildung 1.7c) wird ein 256x256 großes Quadrat so in die Pointcloud gelegt, dass alle selektierten Punkte innerhalb dieses Quadrates liegen und aus dem Quadrat wird das Bild erstellt. Diese Bilder wurden erstellt, um zum einen die Auswirkung der Hintergrundfarbe auf die Klassifizierung als ganzes untersuchen zu können, zum anderen wurde auch die Auswirkung auf einen möglichen Bias-Fehler untersucht. Als erster Schritt in Richtung einer inkrementell lernenden Lösung wurde zunächst die komplette Testdatenbank klassifiziert und die Konfidenz des Ergebnisses gespeichert.(Die Konfidenz ist die Wahrscheinlichkeit, mit der das Ergebnis der Klassifizierung laut Klassifizierer korrekt ist) Anhand dieser Ergebnisse wurde die Datenbank in mehrere Teile gespalten, auf den Teilen, welche mit niedriger Konfidenz klassifiziert wurden, nochmals trainiert und anschließend auf dem Teil, welcher mit der höchsten Konfidenz klassifiziert wurde, evaluiert. Die Ergebnisse und Erfahrungen dieser Untersuchung dienten als Basis für den inkrementellen Algorithmus. Um den inkrementell lernenden Algorithmus zu testen wurden nach Abschluss des Trainingsprozesses die finalen Gewichte gespeichert. Anschließend wurden die Datenbanken mit diesen Gewichten erneut klassifiziert. Für die Auswertung der Ergebnisse wurden Confusion Matrices verwendet, da diese zum einen einen Bias-Fehler aufzeigen und zum anderen eine gute grafische Darstellung bieten. Weiters wurde auch ein Kennwert für jeden Klassifizierungsprozess errechnet, um diese rasch vergleichen zu können. Im Folgenden werden zunächst die entwickelten Methoden und Algorithmen



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Abbildung 1.6: Beispiele der Testdatenbank

(a) (b) (c)

Abbildung 1.7: Beispiel der Testdatenbank mit schwarzem (a), weißem (b) und originalem (c) Hintergrund

vorgestellt. Anschließend werden die Experimente und Ergebnisse präsentiert und zusammengefasst.

2 Methoden

Dieses Kapitel beschäftigt sich mit den hier angewandten Methoden und Algorithmen, soll diese genauer erklären und die gewählten Parameter anführen. Die Ergebnisse, welche zur Wahl der Parameter geführt haben, sind in Kapitel 3 zu finden.

2.1 Neuronale Netze

Im Folgenden soll eine kurze Erklärung zum Aufbau und der Funktionsweise für neuronale Netze durchgeführt werden. Neuronale Netze bestehen aus Neuronen(Abbildung 2.1), welche über mehrere Eingänge und einen Ausgang verfügen. Jedes Neuron verarbeitet Daten an dem gleichen Schema: Die Eingangsdaten werden mit den jeweiligen Gewichtungsfaktoren gewichtet oder multipliziert. Die Summe aus den gewichteten Eingängen wird von der sogenannten Aktivierungsfunktion(z. B. Sigmoidfunktion) verarbeitet und das Ergebnis dieser bildet den Ausgangs des Neurons[1], [2]. Da neuronale Netze aus mehreren Millionen Neuronen bestehen, werden Neuronen, welche ähnliche Berechnungen durchführen, zu sogenannten Layern(Abbildung 2.2) zusammengefasst, um zum einen die Handhabung zu erleichtern, und zum anderen um funktionale Gruppen zu erstellen[1], [2].

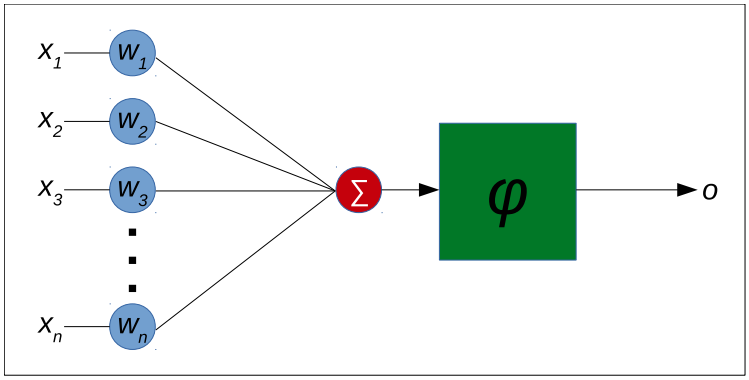


Abbildung 2.1: Schematische Darstellung eines Neurons als Basis eines neuronalen Netzes. Bestehend aus den Eingängen($x_1 - x_n$), welche mit dem jeweiligen Faktor($w_1 - w_n$) gewichtet werden. Die Aktivierungsfunktion(φ) berechnet aus den gewichteten Eingängen den Ausgangswert(o) des Neurons.

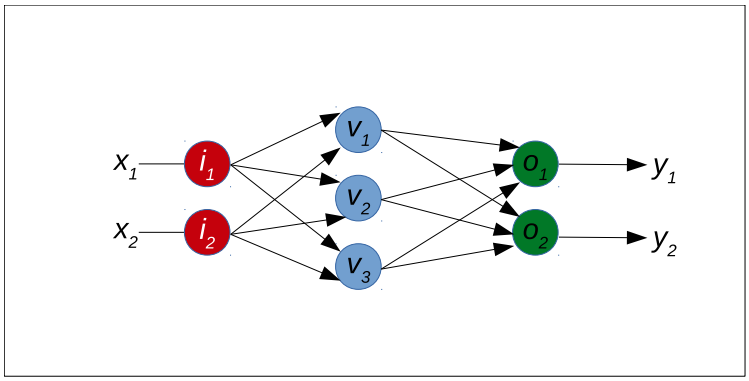


Abbildung 2.2: Beispiel eines neuronalen Netzes bestehend aus drei Layern. Die Eingangsdaten(x_1,x_2) werden an den Input-Layer(i_1,i_2) übergeben. Dieser gibt die Daten an den Hidden-Layer($v_1 - v_3$), welcher die Daten verarbeitet und an den Output-Layer(o_1,o_2) übergibt. Von diesem Layer werden die Ausgänge(y_1,y_2) generiert.

Es gibt viele unterschiedliche Typen von Layern, im Folgenden werden einige angeführt:

- **Pooling-Layer**
  Diese Layer nehmen Maximum, Mittelwert oder andere statistisch signifikante Werte aus Regionen des Vorangegangen Layers und geben diese weiter.z. B. $output = max(inputRegion)$
- **Inner product-Layer**
  Behandelt den Eingang als Vektor und liefert einen Vektor mit einer fixierten Anzahl an Ausgängen.z. B. $output = G \cdot input$ wobei $output \in \mathcal{R}^n$ $G \in \mathcal{R}^{n \times m}$ $input \in \mathcal{R}^m$
- **Convolutional-Layer**
  Diese Layer filtern den Eingang mit unterschiedlichen Kernels und geben die gefilterten Daten weiter.
- **Element wise-Layer**
  Diese Layer führen elementweise mathematische Operationen durch.z. B. $output = input + A$ wobei $output$, $input$, $A \in \mathcal{R}^n$

Diese Art der Berechnung macht eine sehr rasche Verarbeitung möglich, da die Gewichtungsfaktoren und Berechnungsarten komplett in der Hardware implementiert und zur Laufzeit schnell durchgeführt werden können. Für die Entwicklung des Netzes wurde das caffe-Framework[34] verwendet. Dies stellt alle benötigen Funktionen(z. B. Trainieren des Netzes, Verwaltung der Netzdaten, Klassifizierung durch das Netz) zur Verfügung.

Für den Trainingsprozess mit welchem die Gewichte der Layer festgelegt werden wird vom verwendeten Framework Backpropagation[35] mit Stochastic Gradient Descent[36] verwendet. Diese beiden Methoden werden im Folgenden Abschnitt näher beschrieben.

2.2 Trainingsprozess

Alle im Rahmen dieser Arbeit durchgeführten Trainingsprozesse wurden mit Backpropagation[35] und Stochastic Gradient Descent(SGD)[36] durchgeführt.

Bei Backpropagation[35] wird das neuronale Netz zweimal pro Datensatz der Trainingsdatenbank durchlaufen: Zunächst wird vorwärts durch das Netz gerechnet. Hier wird ein Datensatz an den Input-Layer eingegeben und das Ergebnis wie bei einer Klassifikation berechnet. Zusätzlich wird hier auch der Fehler für den Klassifizierungsprozess mittels der Kreuzentropie errechnet. Anschließend wird das korrekte Label hinten an das Netz angelegt und rückwärts

durchgerechnet. Durch diese Berechnung erhält man jene Werte, welche jeder einzelne Knoten liefern müsste um das korrekte Ergebnis zu liefern. So erhält man für jeden Layer Ist- bzw. Soll-Werte für die einzelnen Gewichte und aus diesen kann die Abweichung berechnet werden. Die Gewichte werden anschließend mittels SGD[36] verbessert. Dies geschieht in zwei Schritten zunächst wird der Gesamtfehler der Trainingsdatenbank ermittelt:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i}^{n}(\hat{y}_i - f(x_i,\mathbf{w}) + \lambda r(\mathbf{w})) \tag{2.1}$$

wobei $\lambda r(\mathbf{w})$ den Regularisierungstherm, welcher Overfitting verhindert bezeichnet. Mit diesem Fehler werden nun die Gewichte $\mathbf{w}$ verbessert.

$$\mathbf{v}_{k+1} = \mu\mathbf{v}_k - \alpha\nabla L(\mathbf{w}_k) \tag{2.2}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_t + \mathbf{v}_{k+1} \tag{2.3}$$

Wobei $\mathbf{v}_k$ bzw. $\mathbf{v}_{k+1}$ die Updatewerte der Gewichte zum vorherigen bzw. jetzigen Schritt, μ das sogenannte Momentum, die Auswirkung des vorherigen Updateschrittes, α die Lernrate und $\nabla L(\mathbf{w}_k)$ den Gradienten des Gesamtfehlers bezeichnet. Falls die Trainingsdatenbank sehr viele Daten beinhaltet, wird für den hier beschriebenen Trainingsprozess nicht die gesamte Datenbank herangezogen sondern lediglich ein zufällig gewählter Teil der Datenbank. Daher der Name *Stochastic* Gradient Descent.

Die Parameter, mit welchen der Trainingsprozess beeinflusst werden kann sind Momentum μ und Lernrate α. Von diesen Beiden wurde im Rahmen dieser Arbeit nur die Lernrate variiert. Das Momentum wurde bei allen Trainingsprozessen 0,9 gesetzt. Die Lernrate wurde beim ersten Trainingsprozess zu 0,01 gewählt. Diese beiden Werte werden vom caffe-Framework[34] als gute Startwerte angeführt. Für das inkrementelle Lernen zeigten unsere Untersuchungen, welche in Abschnitt 3.2 zu finden sind, dass eine Lernrate von 0,0001 zu besseren Ergebnissen führt. Diese Lernrate wurde auch für den zur Bias-Überprüfung gehörenden Trainingsprozess gewählt.

2.3 *self-check* Methode zur Bias-Überprüfung und Korrektur

Häufiges Training auf Bildern welche das selbe Objekt zeigen, können zu einem Bias-Fehler führen. Um dies zu verhindern wurde eine Möglichkeit entwickelt, um diesem Fehler entgegenzuwirken. Für diesen Algorithmus wurden aus der Trainingsdatenbank zufällig 20 Bilder pro Objekt(in Summe 200) ausgewählt,

welche vom Klassifizierer mit den Gewichten $\mathbf{w}_0$ korrekt erkannt werden. Nach jedem iterativen Trainingsprozess wird der Klassifizierer auf diesen 200 Bildern evaluiert und sollte eines nicht richtig erkannt werden, wird der Klassifizierer auf allen Bildern erneut trainiert. Dies reduziert die Auswirkung des Bias-Fehlers. Allerdings erfordert das häufige Evaluieren und Trainieren auf den 200 Bildern der Trainingsdatenbank eine Augmentierung der Daten.

2.4 Bild Augmentierung

Häufiges Testen und Evaluieren des iterativ lernenden Algorithmus auf denselben Daten kann zu Overfitting führen. Hierbei handelt es sich um einen Vorgang bei dem der Algorithmus zu genau auf den verwendeten Daten trainiert wird und auf ähnlichen Daten nicht die selbe Performance erreicht. Um diesem Effekt entgegenzuwirken werden die Daten, welche im Rahmen der Bias-Überprüfung verwendet werden, augmentiert. Hierzu wurde eine Funktion entwickelt, welche ein Bild mit einem von elf verschiedenen Algorithmen augmentiert. Die elf Augmentierungsarten, die in dieser Arbeit verwendet werden, sind:

1. Rotation um 90°
2. Rotation um 180°
3. Rotation um 270°
4. Spiegelung vertikal
5. Spiegelung horizontal
6. Leichte Erhöhung der Helligkeit
7. Leichte Verringerung der Helligkeit
8. Starke Erhöhung der Helligkeit
9. Starke Verringerung der Helligkeit
10. Hinzufügen von leichtem Rauschen
11. Hinzufügen von starkem Rauschen

Grafiken zu den einzelnen Arten sind in Abbildung 2.3 mit den jeweiligen Nummern der Aufzählung zu sehen, wobei Abbildung 2.3(12) das originale Bild zeigt.

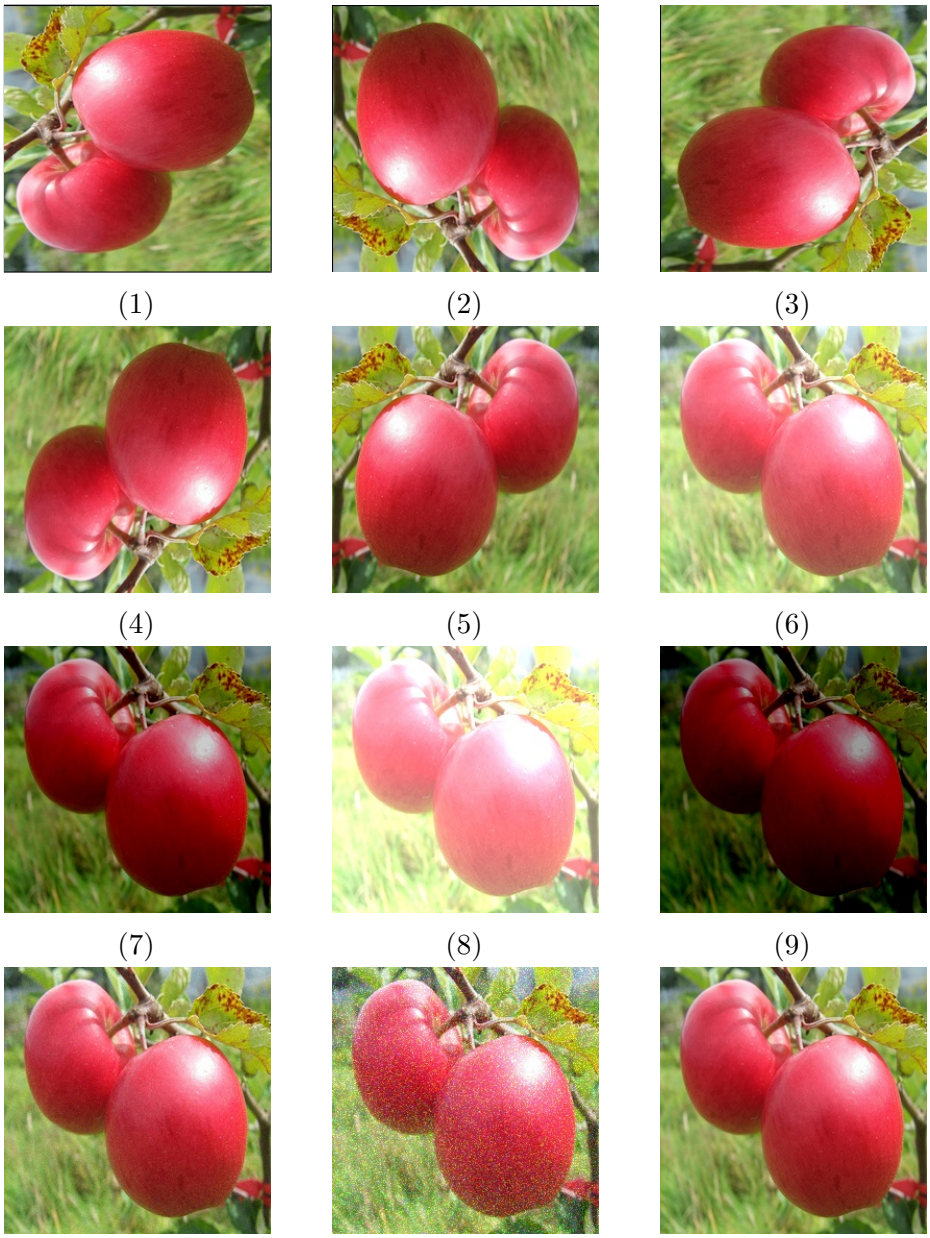


Abbildung 2.3: Die verschiedenen Augmentierungsalgorithmen, wobei (12) das originale Bild zeigt

2.5 Iterativer Algorithmus

In diesem Abschnitt soll der Ablauf des iterativen Lernprozesses näher erklärt werden. Nachdem der Klassifizierer initialisiert, werden die einzelnen zu klassifizierenden Bilder iterativ abgearbeitet:

Zunächst wird das Bild geladen, gegebenenfalls augmentiert und das korrekte Label geladen. Anschließend wird das Bild klassifiziert und die Konfidenz der Klassifizierung überprüft. Sollte diese über dem minimal gewünschten Wert liegen, wird der entsprechende Eintrag in der Confusion Matrix inkrementiert und mit dem nächsten Bild fortgefahren. Falls die Konfidenz nicht hoch genug ist, wird der Klassifizierer auf auf dem Bild trainiert, welches zuvor wenn gewünscht noch augmentiert wurde. Anschließend wird die Bias-Überprüfung wie in Abschnitt 2.3 beschrieben durchgeführt. Nun wird auch hier der entsprechende Eintrag in der Confusion Matrix erhöht und mit dem nächsten Bild fortgefahren. Eine Darstellung des Algorithmus ist in Algorithm 1 zu finden.

Algorithm 1 iterativer Algorithmus

```
for i = 1...#images do
    img ←load(image(i))
    if wanted then
        img ←augment(img)
    end if
    classify(img)
    if prob < minprob then
        if wanted then
            img ←augment(img)
        end if
        train_Classifier_on(img)
        for j = 1..200 do
            bc_img ←load(bc_image(j))
            bc_img ←augment(bc_img)
            classify(bc_img)
            if bc_img classification == wrong then
                bias_error ← true
            end if
        end for
        if bias_error == true then
            train_classifier(bc_images)
        end if
    end if
    increase(CM)
end for
```

2.6 Auswertung der Confusion Matrices

Der entwickelte Algorithmus erstellt sogenannte Confusion Matrices (CM) Abbildung 2.4 zeigt ein Beispiel. Hierbei handelt es sich um quadratische Matrizen, welche für jedes Objekt die korrekte und vom Klassifizierer gewählte Klasse zeigen. In einer Zeile dieser Matrix sind alle Objekte angeführt die als die jeweilige Klasse klassifiziert werden sollen, z. B. alle Objekte die in Zeile 3 eingetragen werden, sollen als Flaschen klassifiziert werden. In den Spalten werden die Objekte so eingetragen wie sie der Algorithmus selbst klassifiziert hat, z. B. alle Objekte die in Spalte 6 eingetragen werden, wurden als Donut erkannt. Der Algorithmus liefert diese Matrizen als Textdokumente zurück. Daher wurden diese Matrizen visuell wie in Abbildung 2.4 aufbereitet, um das Auswerten zu erleichtern. Weiters wurde im Zuge dieser Aufbereitung der zweite für die Auswertung verwendete Parameter, welcher hier als Accuracy(A) bezeichnet wird, berechnet. Die Formel hierfür lautet:

$$A = \frac{\sum_{i=1}^{10} a_{ii}}{\sum_{i=1}^{10} \sum_{j=1}^{10} a_{ij}} \tag{2.4}$$

wobei a_{ij} das Element der Matrix in der i-ten Zeile und j-ten Spalte kennzeichnet. Dieser Wert gibt die Anzahl an korrekt klassifizierten Ergebnissen zur Gesamtanzahl an Klassifizierungen an.

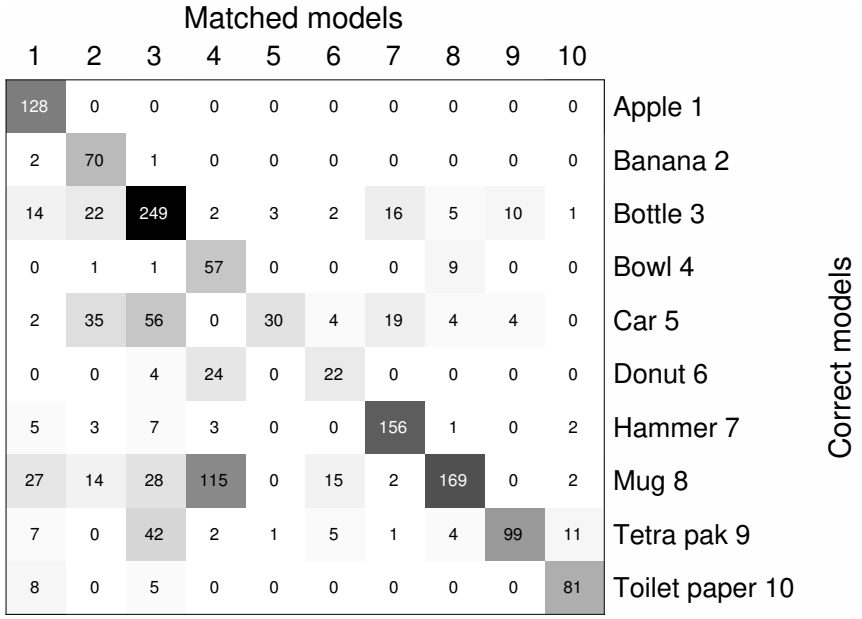


Abbildung 2.4: Beispiel ein Confusion Matrix

3 Experimente und Ergebnisse

Dieses Kapitel soll die einzelnen Evaluierungsprozesse des entwickelten Algorithmus sowohl auf der Testdatenbank als auch der Trainingsdatenbank sowie deren Ergebnisse näher erläutern. Da bei den Evaluierungen häufig zwischen den Hintergrundfarben unterschieden wird, werden die Klassifizierer teilweise als 'schwarzer', 'weißer' oder 'originaler' bezeichnet. Dies bezieht sich auf die Hintergrundfarbe, mit welcher der jeweilige Klassifizierer arbeitet.

3.1 Experimente ohne iteratives Training

Zunächst wurden Untersuchungen ohne jegliches Training auf den Daten der Testdatenbank durchgeführt, um abschätzen zu können, ob der Algorithmus sich verbessert. Weiters konnten so einzelne Teilaspekte(z. B. Augmentierung oder Wahl der Hintergrundfarbe der Bilder) und deren Auswirkung überprüft werden.

3.1.1 Klassifizierung der gesamten Testdatenbank

Diese Untersuchung dient als Basiswert für die Klassifizierung. Hier wurde das Convolutional Neural Network (CNN) lediglich mit den Gewichten aus AlexNet[3] geladen und der letzte Layer auf der Trainingsdatenbank trainiert. Anschließend wurden der Reihe nach alle Bilder der Testdatenbank klassifiziert und die Ergebnisse als Confusion Matrices (CMs) dargestellt. Insgesamt wurden pro Klassifizierung 5 verschiedene CMs erstellt: Eine enthält alle Klassifizierungen, welche mit einer Konfidenz von unter 80% klassifiziert wurden, die nächste enthält jene mit einer Konfidenz von 80% oder höher. Die nächsten beiden enthalten jene mit einer Konfidenz von unter bzw. über und 90% und die letzte enthält alle Klassifizierungen unabhängig von der jeweiligen Konfidenz.
Hier soll neben der Erstellung des Basiswertes für den späteren Vergleich auch die Auswirkung des Hintergrundes auf die Klassifizierung überprüft werden. Abbildung 3.1.1.1 zeigt die CMs, welche alle Klassifizierungen enthalten. In den Grafiken ist erkenntlich, dass der Algorithmus mit dem originalem Hintergrund die meisten Bilder korrekt klassifiziert.

Dies zeigt sich auch in der Accuracy(A):

- **Schwarzer Hintergrund:** 57,75% korrekt
- **Originaler Hintergrund:** 65,81% korrekt
- **Weißer Hintergrund:** 54,34% korrekt

$$A = \frac{\#korrekteKlassifizierungen}{\#gesamteKlassifizierungen} \tag{3.1}$$

Abbildung 3.1.1.4 und 3.1.1.5 zeigen die CMs für alle drei Hintergründe mit unter bzw. über 80% Konfidenz und unter bzw. über 90% Konfidenz. Hier sind die Diagonalen in den einzelnen Grafiken vor allem in den Grafiken mit höher Konfidenz(mehr als 80 bzw. 90%) deutlich stärker ausgeprägt. Diese Diagonale repräsentiert die korrekten Klassifizierungen, daher sind nur Einträge in dieser Diagonale gewünscht. Auch hier liefern die Bilder mit dem originalem Hintergrund die besten Ergebnisse(siehe Tabelle 3.1).

Hintergrundfarbe	$< 80\%$	$\geqq 80\%$	$< 90\%$	$\geqq 90\%$
schwarz	26,85%	66,32%	29,53%	70,11%
original	32,94%	74,70%	35,19%	78,27%
weiß	24,63%	62,31%	27,65%	65,69%

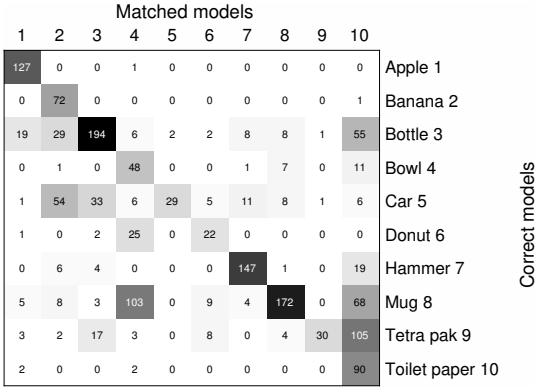
Tabelle 3.1: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Interessant an diesem Ergebnis ist die Verwechslung von Tasse und Schüssel, welche sowohl mit schwarzem als auch weißem Hintergrund auftritt. Der Grund hierfür liegt in der Tatsache der Segmentierung mittels Clustern, wobei lediglich der Cluster, welcher den geringsten euklidischen Abstand ausweist, gewählt wird. Deshalb wird lediglich ein Teil des Objekts dargestellt und so die Klassifizierung erschwert. In Abbildung 3.1.1.2 ist eine Schüssel (a, c) und eine Tasse (b, d) zu sehen. Bei den Bildern mit schwarzem Hintergrund sehen sich die Objekte deutlich ähnlicher als bei jenen mit originalem Hintergrund. Hier könnte eine Anpassung der Parameter der Segmentierung die Ergebnisse verbessern, da die Untersuchung der Segmentierung allerdings nicht Teil dieser Arbeit ist, wurde dies hier nicht untersucht. Interessant ist, dass die Verwechslung von Tasse und Schüssel mit sehr hoher Konfidenz, durch den Klassifizierer passiert (Abbildung 3.1.1.4b&f bzw. Abbildung 3.1.1.5b&f). Dies bedeutet, dass es besser ist, mehr vom Objekt darzustellen als nur das Ergebnis der Segmentierung.

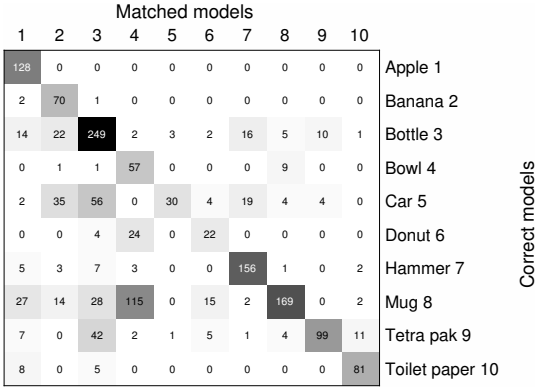
Weiters werden die Bilder von Objekt 5(Auto) und Objekt 6(Donut) mit sämtlichen Hintergründen schlecht erkannt, allerdings ist hier im Fall von Objekt 5 keine klare Verwechslung erkennbar bzw. bei Objekt 6 wird das Objekt nicht korrekt erkannt. Die nach Konfidenzen aufgespaltenen Ergebnisse zeigen, dass der Algorithmus sich bei Objekt 5 allgemein sehr unsicher ist. Dies ist auf die Unterschiede zwischen Trainings- und Testdatenbank zurückzuführen(siehe Abbildung 3.1.1.3). Bei Objekt 6 handelt es sich um eine Mischung der oben genannten Fehler. Bei schwarzem und weißem Hintergrund ist das Objekt selbst schlecht zu erkennen, bei originalem Hintergrund zeigen die Bilder der Trainingsdatenbank zu wenig Ähnlichkeit mit den Bildern der Testdatenbank.

Objekt 10 (Toiletten Papier) weißt bei weißem und schwarzem Hintergrund einen deutlichen Bias auf. Dies ist erneut auf Unterschiede in der Test- und Trainingsdatenbank zurückzuführen. Im Unterschied zu Objekt 5 sind sich die Klassifizierer hier sicherer bei den Klassifizierungen. Dieser Fehler ist auf die Bilder der Trainingsdatenbank, welche neben Objekt 10 auch andere Objekte zeigen, zurückzuführen. Durch diese anderen Objekte spielt hier die Hintergrundfarbe eine deutliche Rolle.

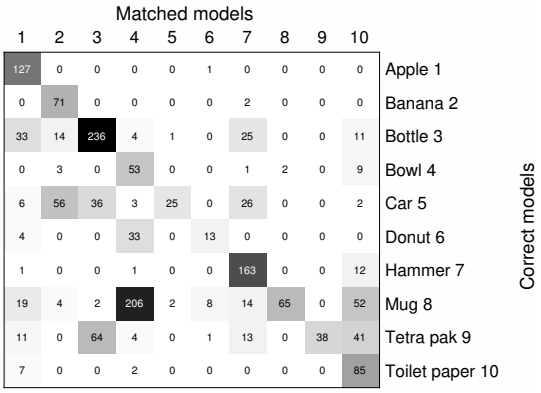
Weiters weisen die Klassifizierer bei allen drei Hintergrundfarben einen Bias bei Objekt 3(Flasche) und Objekt 4(Schüssel) auf. Diese werden in den weiteren Untersuchungen wenn benötigt als grundlegender Bias bezeichnet.



(a) schwarzer Hintergrund

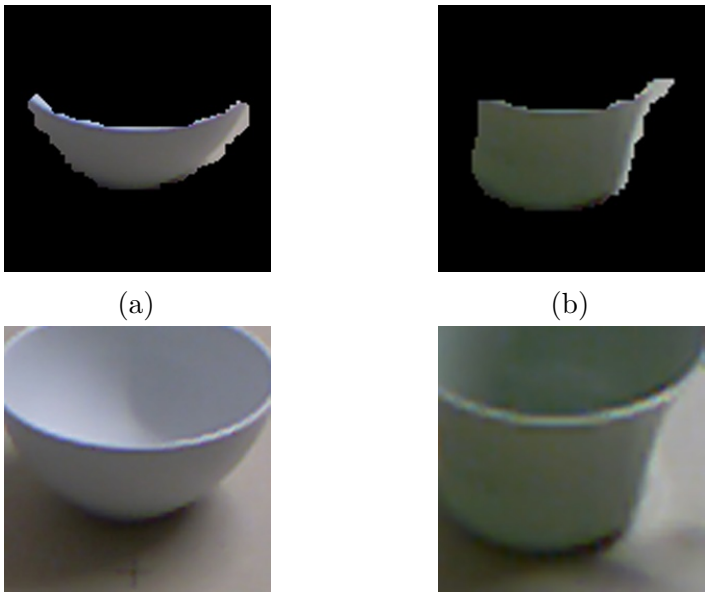


(b) originaler Hintergrund



(c) weißer Hintergrund

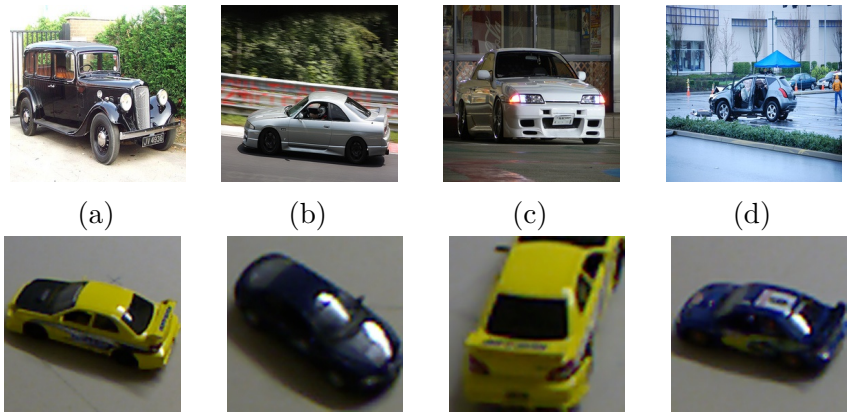
Abbildung 3.1.1.1: Confusion Matrices der unterschiedlichen Hintergründe ohne inkrementelles Lernen(Alle Klassifizierungen).



(a) (b)

(c) (d)

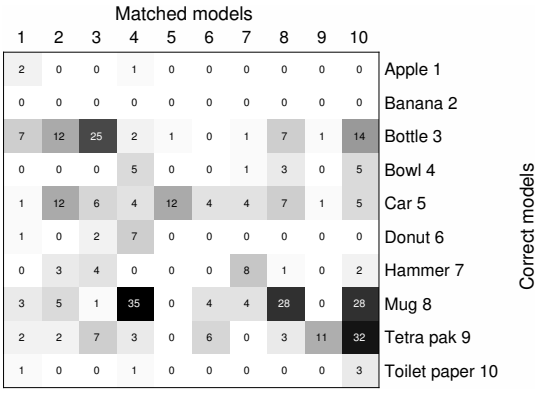
Abbildung 3.1.1.2: Vergleich zweier Bilder mit schwarzem(a,b) und originalem(c,d) Hintergrund



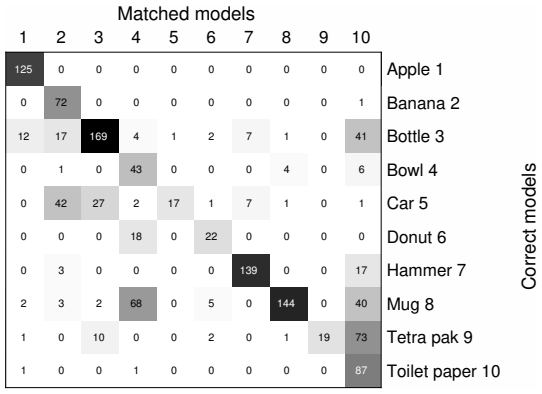
(a) (b) (c) (d)

(e) (f) (g) (h)

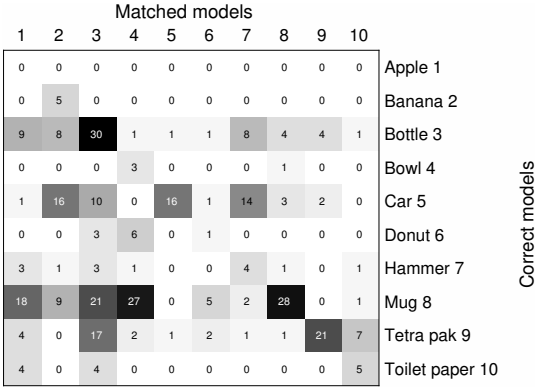
Abbildung 3.1.1.3: Vergleich der Bilder von Objekt 5(Auto) der Trainings(a,b,c,d) und der Testdatenbank(e,f,g,h)



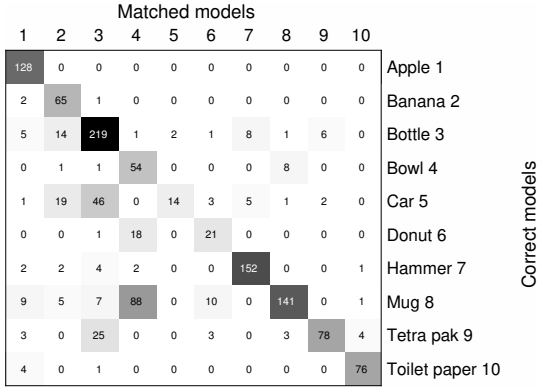
(a) unter 80% schwarzer Hintergrund



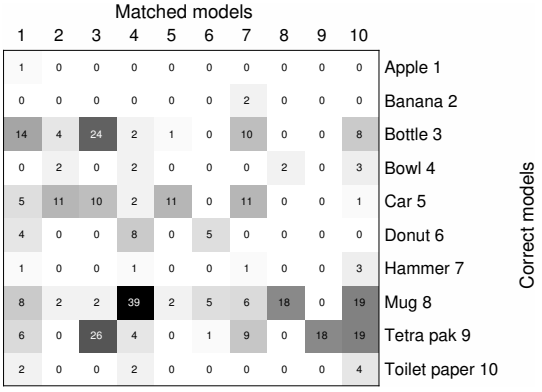
(b) über 80% schwarzer Hintergrund



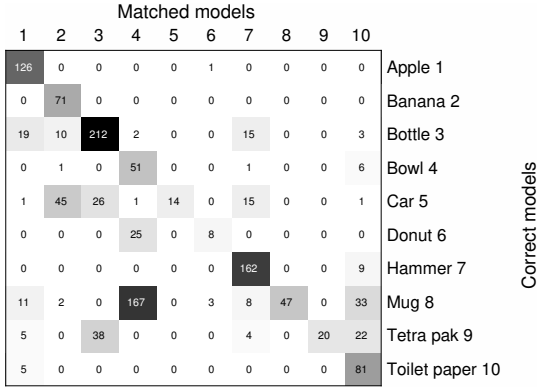
(c) unter 80% originaler Hintergrund



(d) über 80% originaler Hintergrund

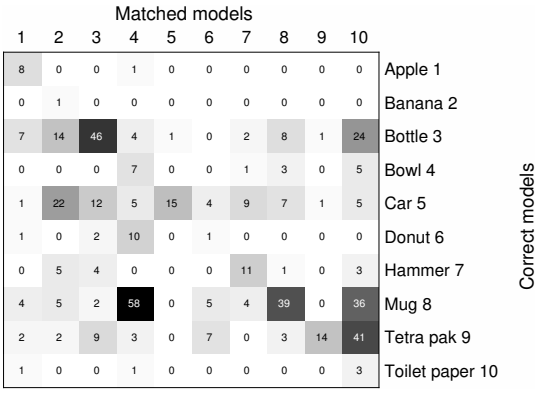


(e) unter 80% weißer Hintergrund

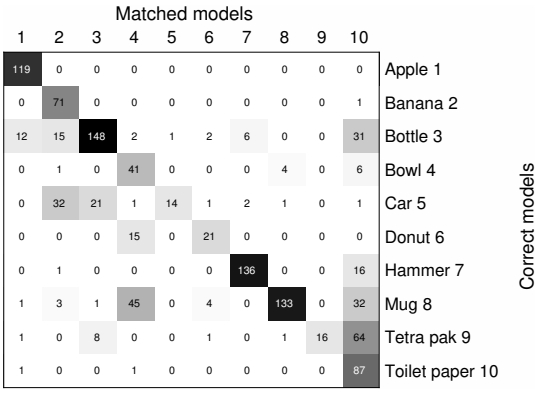


(f) über 80% weißer Hintergrund

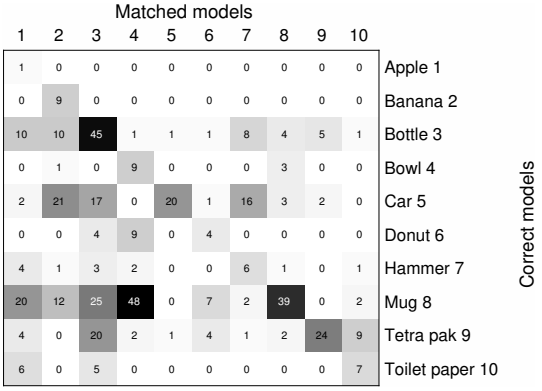
Abbildung 3.1.1.4: Confusion Matrices der unterschiedlichen Hintergründe ohne inkrementelles Lernen(über bzw. unter 80%).



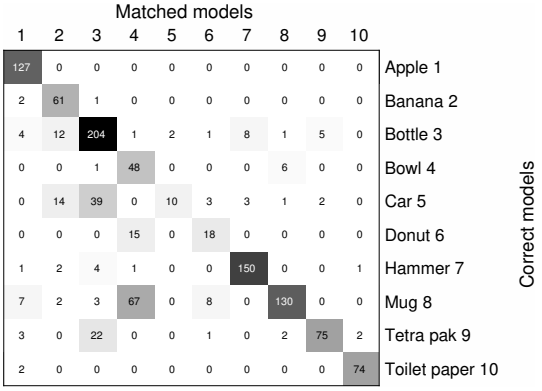
(a) unter 90% schwarzer Hintergrund



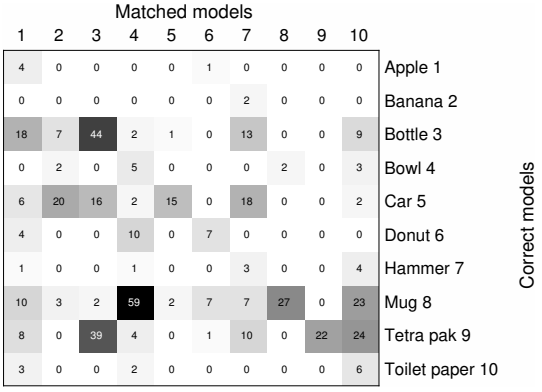
(b) über 90% schwarzer Hintergrund



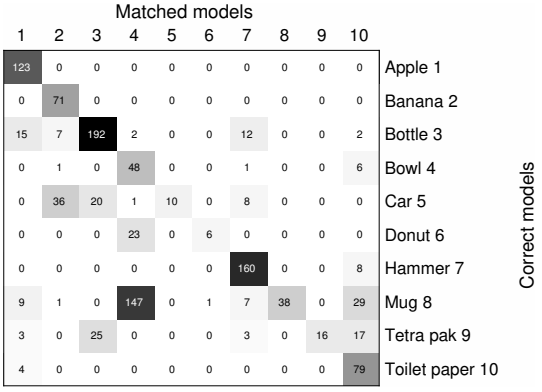
(c) unter 90% originaler Hintergrund



(d) über 90% originaler Hintergrund



(e) unter 90% weißer Hintergrund



(f) über 90% weißer Hintergrund

Abbildung 3.1.1.5: Confusion Matrices der unterschiedlichen Hintergründe ohne inkrementelles Lernen(über bzw. unter 90%).

Auf Basis dieser Ergebnisse wurden die nächsten Untersuchungen durchgeführt, welche einen zweiten Trainingsprozess vorsahen um den Klassifizierer an Bilder der Testdatenbank zu gewöhnen. Diese sind in Abschnitt 3.2 zu finden.

3.1.2 Klassifizierung der Trainingsdatenbank

Diese Untersuchung wurde durchgeführt, um die Augmentierung von Bilder evaluieren zu können. Hierzu wurde zunächst die Trainingsdatenbank, also jene Bilder welche zum Training des Klassifizierers verwendet wurden, direkt klassifiziert, um auch hier Ausgangswerte zu erhalten. Das Ergebnis dieses Tests ist in Abbildung 3.1.2.1 zu sehen. Die Diagonale ist deutlich zu erkennen, was für eine gute Klassifizierung spricht. Die gesamte Accuracy beträgt 87,31%.

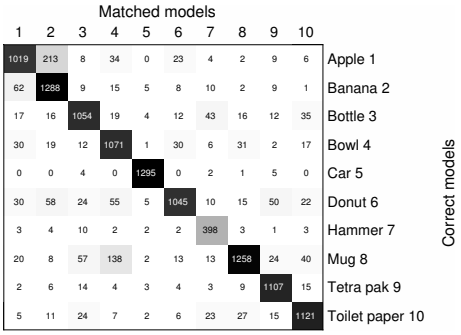


Abbildung 3.1.2.1: Confusion Matrices der Klassifizierung der Trainingsdatenbank

Im folgenden werden die Augemtierungsmethoden getestet. Hier wurde der Klassifizierer geladen, alle Bilder der Trainingsdatenbank mit der jeweiligen Methode augmentiert und der Klassifizierer auf diesen evaluiert. Diese Methoden wurden getestet, Beispielbilder sind in Abbildung 2.3 zu sehen:

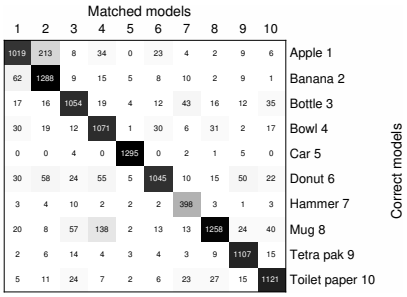
- Rotationen
- Spiegelungen
- Helligkeitsveränderungen
- Rauschen

Rotations Augmentierung

Die drei Rotationsaugmentierungen(90, 180 und 270°) liefern unterschiedliche Ergebnisse(siehe Abbildung 3.1.2.2). Die gesamte Accuracy beträgt 63,83%

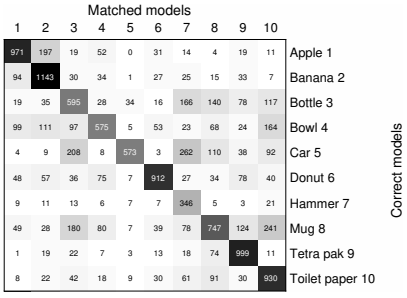
bei 90° Rotation, 69,94% bei 180° Rotation und 63,41% bei 270° Rotation. Allgemein ist die Klassifizierung bei gedrehten Bildern schlechter. Lediglich bei der Rotation um 180° sieht man ein wenig bessere Ergebnisse als bei den anderen. Dies ergibt sich aus der Tatsache, dass die hier verwendeten Objekte, sie eine bevorzugte Ausrichtung besitzen, diese eher um 180° verändern als um 90 bzw. 270°. Ein Beispiel hierfür wäre die Tasse, welche eher auf dem Kopf steht als seitlich zu liegen.

(a) original



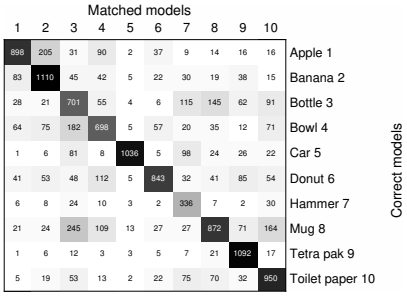
(b) original $A = 87{,}31\%$

(c) 90°



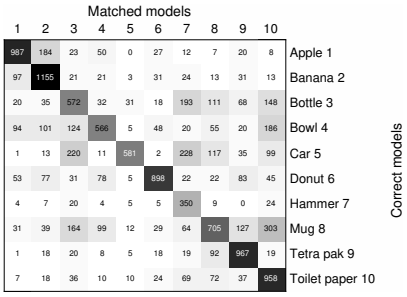
(d) 90° $A = 63{,}83\%$

(e) 180°



(f) 180° $A = 69{,}94\%$

(g) 270°



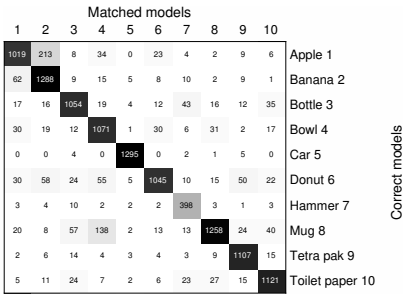
(h) 270° $A = 63{,}41\%$

Abbildung 3.1.2.2: Confusion Matrices der unterschiedlichen Rotations-Augmentierungen.

Spiegelungs Augmentierung

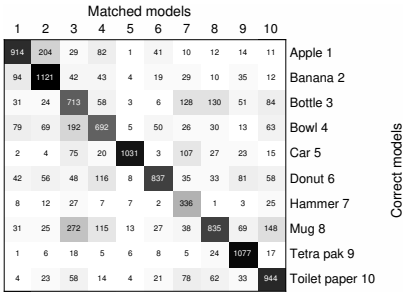
Die zwei Spiegelungs-Augmentierungen(vertikal und horizontal) liefern unterschiedliche Ergebnisse(siehe Abbildung 3.1.2.3). Die gesamte Accuracy beträgt 69,64% bei vertikaler Spiegelung, also um eine horizontale Achse, und 87,33% bei horizontaler Spiegelung, also um eine vertikale Achse. Es ist sowohl in der Accuracy als auch in der CM deutlich ersichtlich, dass die horizontale Spiegelung besser wiedererkannt wird.

(a) original



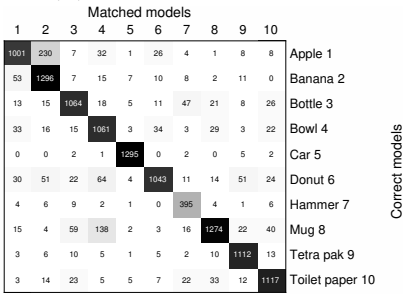
(b) original

(c) vertikale Spiegelung



(d) vertikale Spiegelung

(e) horizontale Spiegelung



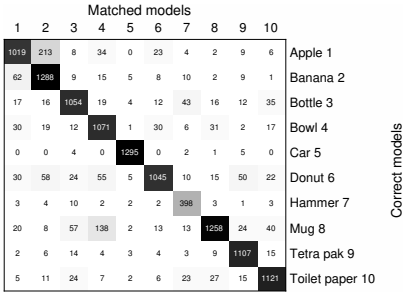
(f) horizontale Spiegelung

Abbildung 3.1.2.3: Confusion Matrices der unterschiedlichen Spiegelungs-Augmentierungen.

Helligkeits Augmentierung

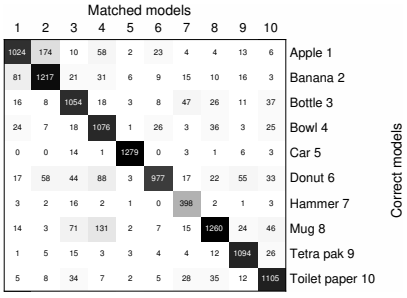
Bei diesen Augmentierungen werden die Farbwerte aller Pixel der Bilder gleich verändert. Die RGB-Werte werden entweder um 50 oder 100 erhöht(heller) oder verringert(dünkler). Die Augmentierungen liefern ähnliche Ergebnisse, wobei die leichtere Veränderung besser erkannt wird(siehe Abbildung 3.1.2.4). Die gesamte Accuracy beträgt 85,90% bei leicht hellerem Bild, 86,33% bei leicht dunklerem Bild, 80,56% bei stark hellerem Bild und 81,30% bei stark dunklerem Bild.

(a) original



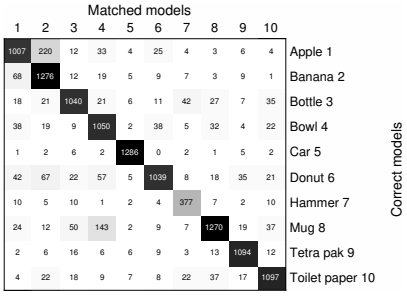
(b) original

(c) leicht heller



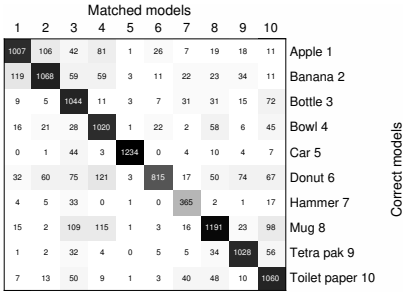
(d) leicht heller

(e) leicht dünkler



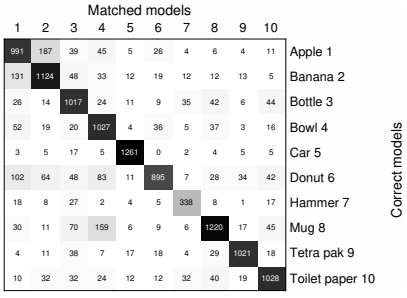
(f) leicht dünkler

(g) stark heller



(h) stark heller

(i) stark dünkler



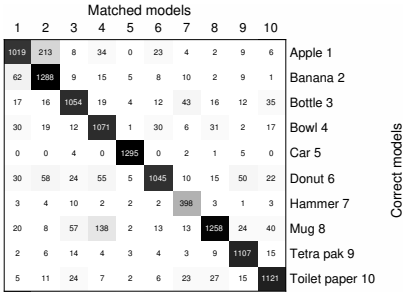
(j) stark dünkler

Abbildung 3.1.2.4: Confusion Matrices der unterschiedlichen Helligkeits-Augmentierungen.

Rausch Augmentierung

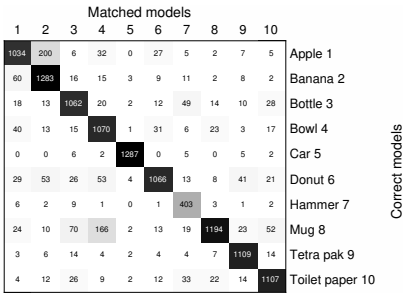
Bei den Rausch Augmentierungen werden die Farbkanäle der Bilder unterschiedlich mit gaußschem Rauschen beeinflusst. Der Mittelwert für das Rauschen ist immer 0, die Standartabweichung beträgt 10 bei leichtem und 50 bei starkem Rauschen. Die Augmentierungen liefern unterschiedliche Ergebnisse(siehe Abbildung 3.1.2.5). Die gesamte Accuracy beträgt 86,97% bei leichtem Rauschen und 73,23% bei starkem Rauschen. Es ist sowohl in der Accuracy als auch in der CM deutlich ersichtlich, dass geringes Rauschen besser wiedererkannt wird. Allerdings ist ein Unterschied zu keinem Rauschen ersichtlich.

(a) original



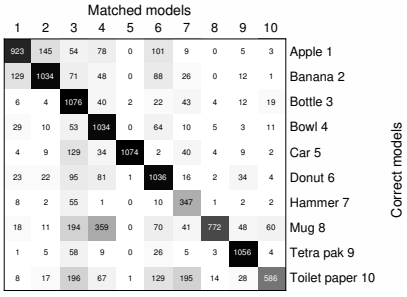
(b) original

(c) leichtes Rauschen



(d) leichtes Rauschen

(e) starkes Rauschen



(f) starkes Rauschen

Abbildung 3.1.2.5: Confusion Matrices der unterschiedlichen Rausch-Augmentierungen.

3.2 Klassifizierungen mit zweitem Trainingsprozess

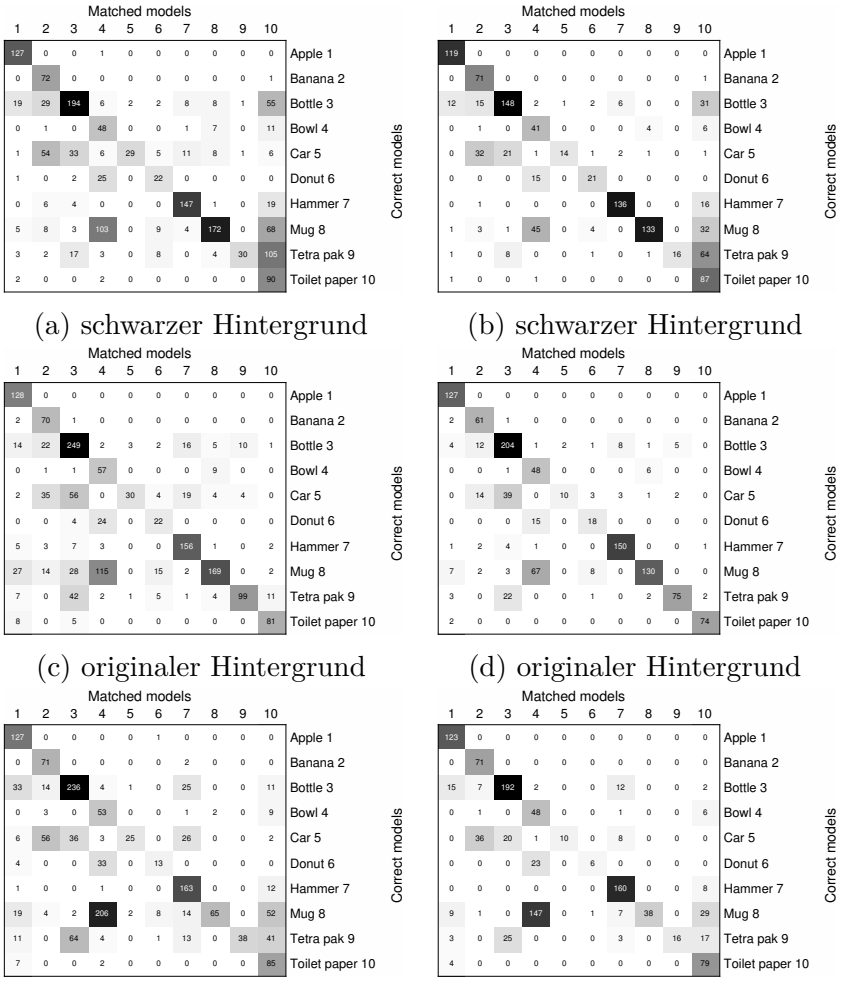
Diese Untersuchungen dienten zum einen zur Überprüfung ob iteratives Training möglich ist, zum anderen, wie die Lernrate sowie die Trainingszahl zu wählen sind, damit eine Verbesserung des Algorithmus auftritt aber ein Bias gering bleibt.

3.2.1 Erweiterung der Trainingsdatenbank

Die Untersuchungen dieses Abschnittes beschäftigen sich mit der Verbesserung des Klassifizerers durch einen zweiten Trainingsprozess auf Daten die zuvor vom Klassifizerers nicht sicher genug klassifiziert wurden. Hierzu wurde die Testdatenbank auf Basis der Ergebnisse von Abschnitt 3.1.1 für jede der drei Bildtypen(schwarz,original,weiß) in drei unterschiedliche Teile geteilt:

1. Jene Bilder welche mit einer Konfidenz von 90% oder höher klassifiziert wurden
2. Jene Bilder welche mit einer Konfidenz unter 80% klassifiziert wurden
3. Jene Bilder welche mit einer Konfidenz unter 90% klassifiziert wurden

Der erste Teil diente als neues Testset und die beiden anderen als Trainingssets. Für den zweiten Trainingsprozess wurde hier eine Lernrate von 0,01 bei einer Trainingszahl von 100 gewählt. Diese Werte wurden auch für das erste Training auf dem Trainingsset gewählt. Um erneut Vergleichswerte zu erhalten wurden zunächst die verkleinerten Testsets ohne zweitem Trainingsprozess klassifiziert. Die Ergebnisse dieser Klassifizierung ist in Abbildung 3.2.1.1 dargestellt. Die Accuracy beträgt 70,11% bei schwarzem Hintergrund, 78,27% bei originalem Hintergrund und 65,69% bei weißem Hintergrund.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

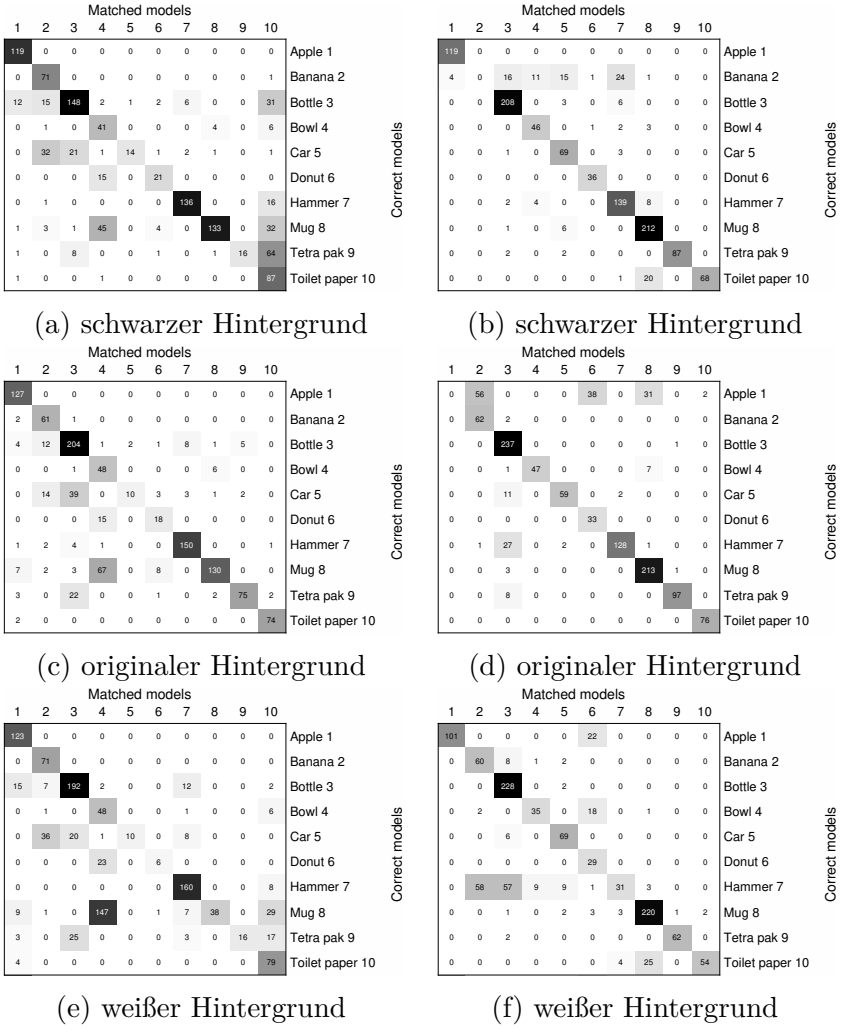
Abbildung 3.2.1.1: Confusion Matrices der unterschiedlichen vollen(links) bzw. verkleinerten Testsets(rechts).

Training bei einer Konfidenz unter 80%

Abbildung 3.2.1.2 zeigt die CMs für die unterschiedlichen Hintergründe bei einem zusätzlich Training auf dem zweiten oben genannten Teil. Dieser Teil besteht aus jenen Bildern, welche zuvor vom CNN nur mit einer Konfidenz von unter 80% klassifiziert wurden. Anschließend wurde der Klassifizierer auf dem jeweiligen verkleinerten Testset evaluiert. Die Accuracy beträgt 87,77% bei schwarzem Hintergrund, 83,70% bei originalem Hintergrund und 78,60% bei weißem Hintergrund. Bei allen Hintergrundfarben ist eine deutliche Verbesserung in der CM und der Accuracy ersichtlich. Allerdings verlernt der Klassifizierer bei allen 3 Hintergründen Objekte. Beim schwarzen Hintergrund verlernt er die Banane, beim originalen Hintergrund den Apfel und beim weißen Hintergrund den Hammer.

Training bei einer Konfidenz unter 90%

Abbildung 3.2.1.2 zeigt die CM für die unterschiedlichen Hintergründe bei einem zusätzlich Training auf dem dritten oben genannten Teil, jene Bilder, welche zuvor vom CNN nur mit einer Konfidenz von unter 90% klassifiziert wurden. Anschließend wurde der Klassifizierer auf dem jeweiligen verkleinerten Testset evaluiert. Die Accuracy beträgt 93,93% bei schwarzem Hintergrund, 90,22% bei originalem Hintergrund und 84,96% bei weißem Hintergrund. Hier ist bei allen 3 Hintergrundarten eine deutliche Verbesserung der Accuracy sowie der CMs sichtbar. Lediglich vereinzelte Objekte weisen Verwechslungen auf, beispielsweise Objekt 10 bei schwarzem Hintergrund und Objekt 1 bei originalem Hintergrund. Objekt 7 bei weißem Hintergrund kann auch hier nicht komplett richtig klassifiziert werden. Diese Verwechslungen treten Aufgrund des hohen Anteils an weißen Pixeln in den Bildern, wegen des langen dünnen Stiels des Hammers auf.

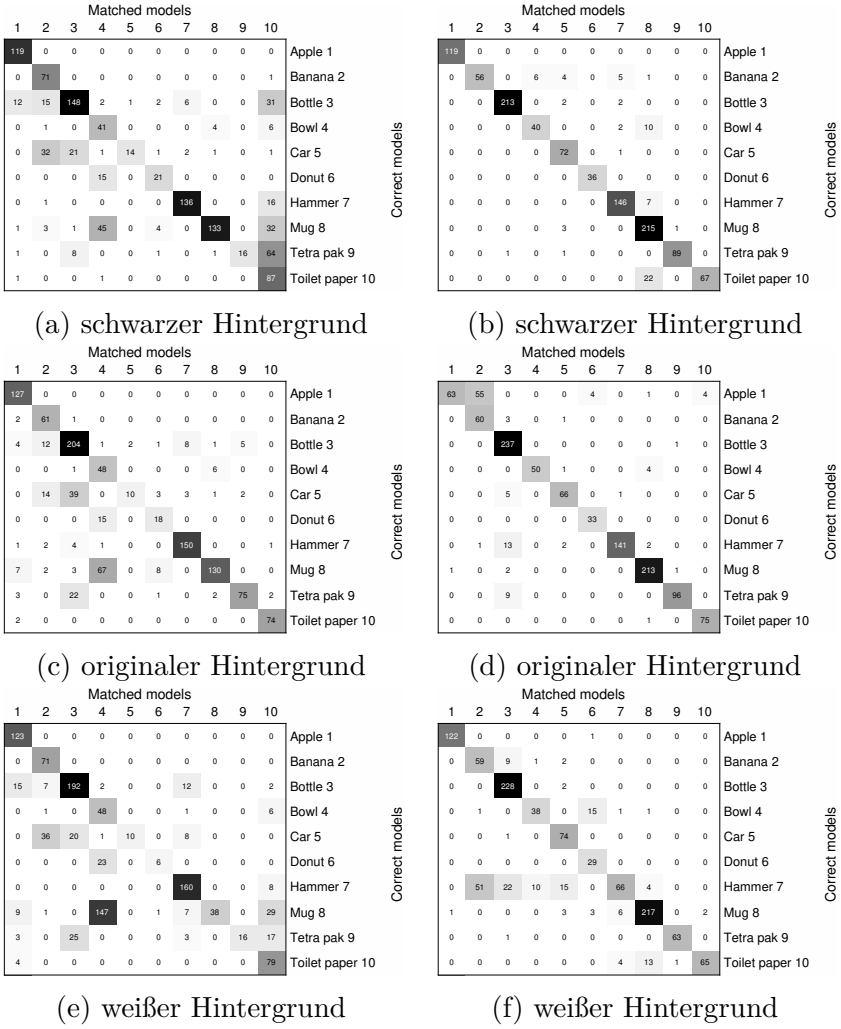


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.2.1.2: Confusion Matrices der unterschiedlichen verkleinerten Testsets vor(links) bzw. nach(rechts) Training auf Daten unter 80% Konfidenz.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.2.1.3: Confusion Matrices der unterschiedlichen verkleinerten Testsets vor(links) bzw. nach(rechts) Training auf Daten unter 90% Konfidenz.

3.2.2 Untersuchung der Lernrate und Trainingszahl

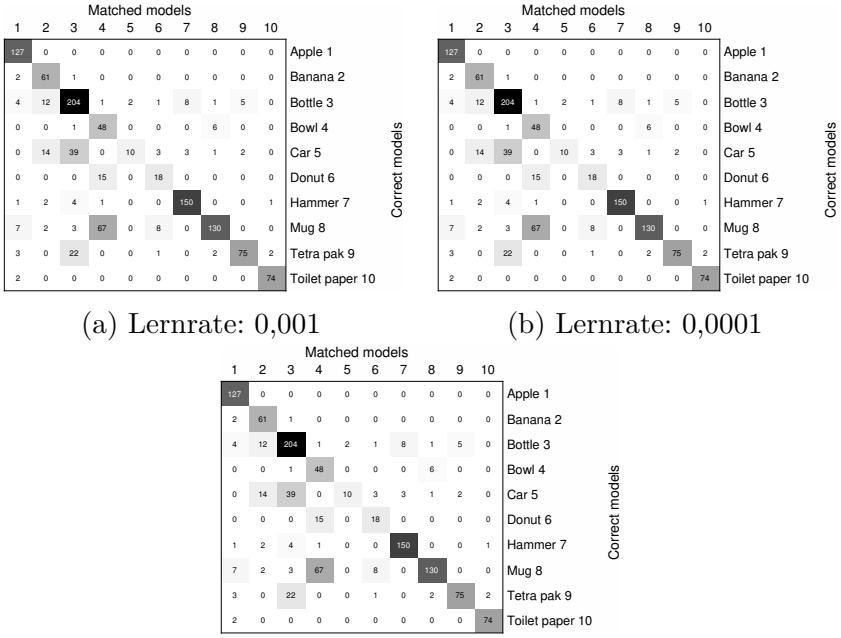
Beide hier untersuchten Parameter spielen beim iterativen Trainieren des Algorithmus eine große Rolle. Die Lernrate bestimmt wie stark die Gewichte der einzelnen Layer bei jedem Lernschritt verändert werden. Die Trainingszahl bestimmt, wie viele Lernschritte bei einem Trainingsprozess durchgeführt werden. Es wurden 3 unterschiedliche Werte für Lernrate (0,001 0,0001 0,00001) und Trainingszahl (1 10 100) gewählt und in unterschiedlichen Kombinationen getestet. Die Tests wurden mit Bildern der Testdatenbank mit originalem Hintergrund durchgeführt. Der Algorithmus wurde auf einem Bild, mit den oben angeführten Parametern, trainiert und anschließend auf dem verkleinerten Testset evaluiert. Es wurden Test mit den beiden Klassen durchgeführt, welche die größte Anzahl an Klassifizierungen unter 90% aufweisen.

Training auf einer Flasche

Bei dem Trainieren auf dem Bild einer Flasche ist bei einer Trainingszahl von 1(Abbildung 3.2.2.1) keinerlei Veränderung der CM egal bei welcher Lernrate ersichtlich. Dies ist auf die Tatsache zurückzuführen, dass der Algorithmus bei den gewählten Lernraten mehr als einen Schritt benötigt um die Gewichte stark genug verändern zu können.

Bei einer Trainingszahl von 10(Abbildung 3.2.2.2) ist die Auswirkung der Lernrate deutlich ersichtlich: Bei einer Lernrate von 0,001 ist eine deutliche Verzerrung des CNNs ersichtlich, welche bei einer Lernrate von 0,0001 deutlich abgeschwächt auftritt. Allerdings muss hierbei berücksichtigt werden, dass der Klassifizierer auch ohne zusätzliche Training bereits leicht verzerrt ist, da einige Objekte häufiger als Flaschen klassifiziert werden. Bei einer Lernrate von 0,00001 ist hier erneut keine Veränderung er CM ersichtlich, da die Lernrate zu gering ist.

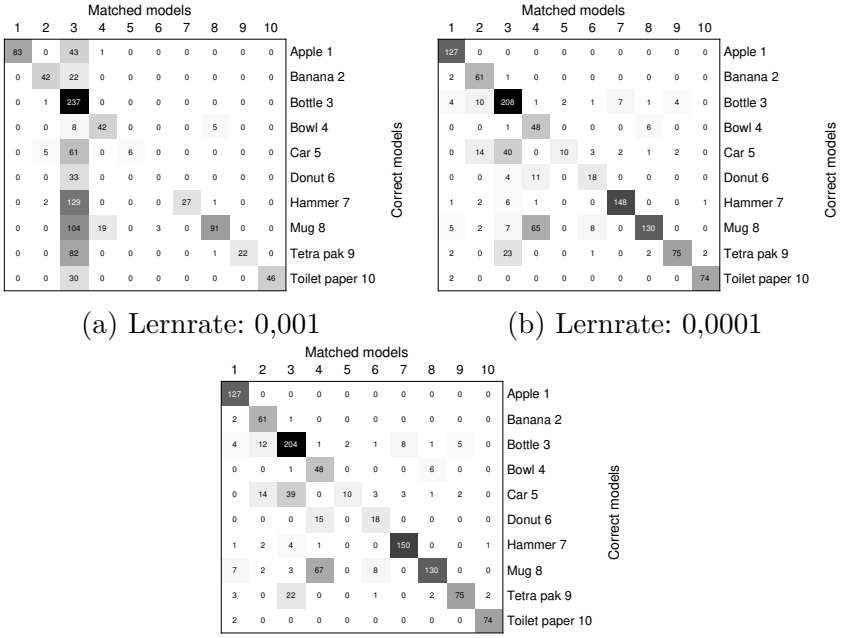
Bei einer Trainingszahl von 100(Abbildung 3.2.2.3) ist die Verzerrung bei einer Lernrate von 0,001 so stark, dass nahezu alle Objekte als Flaschen klassifiziert werden. Bei einer Lernrate von 0,0001 ist die Verzerrung stärker ersichtlich als bei vorherigen Trainingszahl und bei einer Lernrate von 0,00001 zeigen sich ähnliche Ergebnisse wie bei einer Lernrate von 0,0001 mit einer Trainingszahl von 10.



(a) Lernrate: 0,001 (b) Lernrate: 0,0001

(c) Lernrate: 0,00001

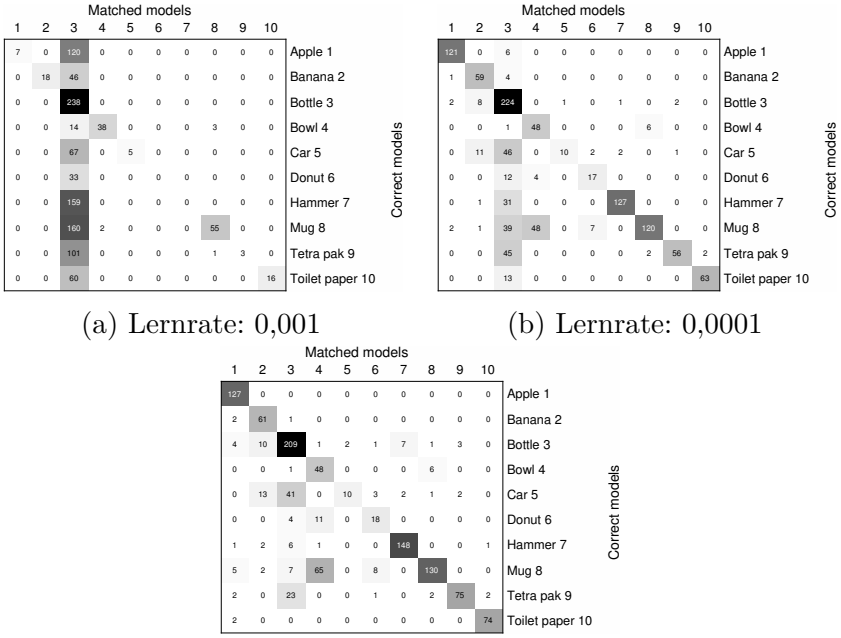
Abbildung 3.2.2.1: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Flasche bei einer Trainingszahl von 1.



(a) Lernrate: 0,001 (b) Lernrate: 0,0001

(c) Lernrate: 0,00001

Abbildung 3.2.2.2: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Flasche bei einer Trainingszahl von 10.



(a) Lernrate: 0,001 (b) Lernrate: 0,0001

(c) Lernrate: 0,00001

Abbildung 3.2.2.3: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Flasche bei einer Trainingszahl von 100.

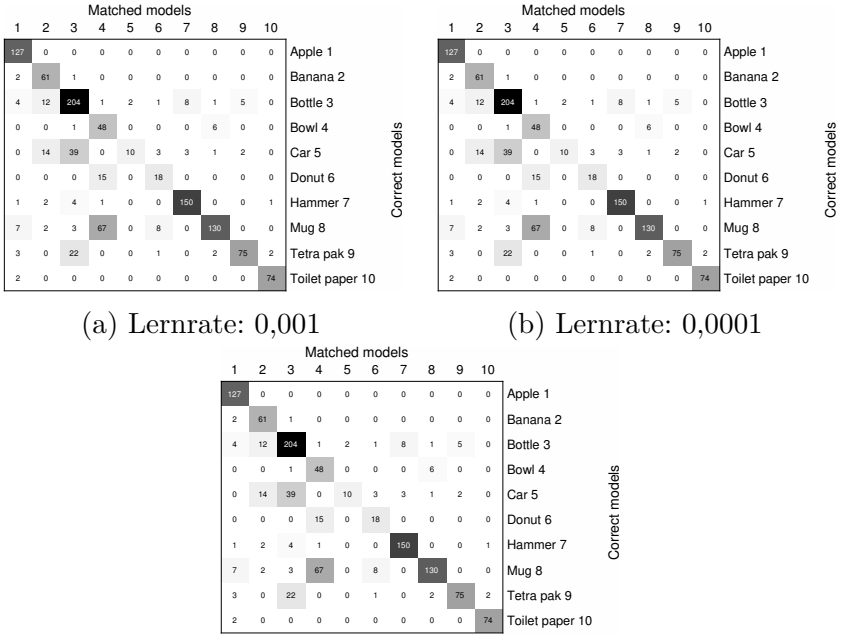
Training auf einer Tasse

Auch bei Training auf einem Bild einer Tasse ist bei einer Trainingszahl von 1(Abbildung 3.2.2.4) keinerlei Veränderung der CMs zu erkennen. Auch hier ist die Trainingszahl zu gering, als dass eine Veränderung erreicht werden könnte.

Bei einer Trainingszahl von 10(Abbildung 3.2.2.5) sind die Ergebnisse erneut denen des Trainings mit dem Bild einer Flasche ähnlich. Bei einer Lernrate von 0,001 ist ein sehr deutlicher Bias bei Objekt 3 zu erkennen. Bei einer Lernrate von 0,0001 ist ein leichter Bias zu erkennen, allerdings erhöht die Zahl der korrekt erkannten Tassen von 130 auf 168 erhöht werden. Bei einer Lernrate von 0,00001 ist erneut keine Veränderung zu erkennen, da die Lernrate zu gering ist.

Bei einer Trainingszahl von 100(Abbildung 3.2.2.6) ist der Bias bei einer Lernrate von 0,001 noch deutlicher ausgeprägt. Fast alle Objekte werden als Tasse klassifiziert. Auch bei einer Lernrate von 0,0001 ist der Bias stärker ausgeprägt, als bei Training mit einer niedrigeren Lernrate. Auch hier ist bei einer Lernrate von 0,00001 erneut eine Verbesserung der Klassifizierung der Tasse, sowie ein leichter Bias zu erkennen.

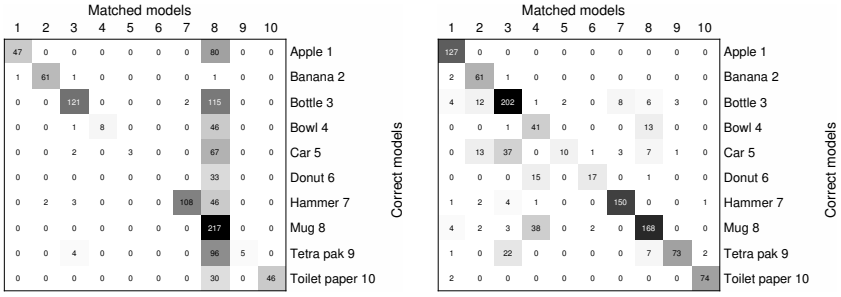
Aus den Ergebnissen dieser beiden Untersuchungen wurde für die iterativen Trainingsprozesse eine Lernrate von 0,0001 bei einer Trainingszahl von 10 gewählt. Bei diesen Parametern weisen beide Untersuchungen eine Verbesserung der Klassifizierung, bei einem geringen Bias auf. Die andere Parameterkombination, welche aufgrund der Ergebnisse in Frage gekommen wäre(Lernrate 0,00001 Trainingszahl 100) wurde aufgrund der hohen Trainingszahl und damit verbundenen längeren Dauer des Trainingsprozesses nicht gewählt. Für Training im Rahmen von *self-check* wurden dieselben gewählt, da die Korrektur als iteratives Training angesehen wurde.



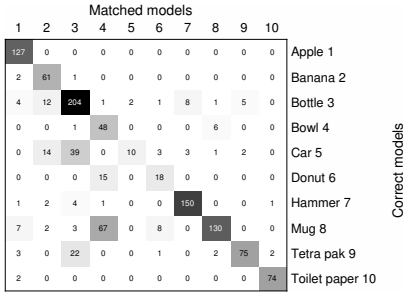
(a) Lernrate: 0,001 (b) Lernrate: 0,0001

(c) Lernrate: 0,00001

Abbildung 3.2.2.4: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Tasse bei einer Trainingszahl von 1.

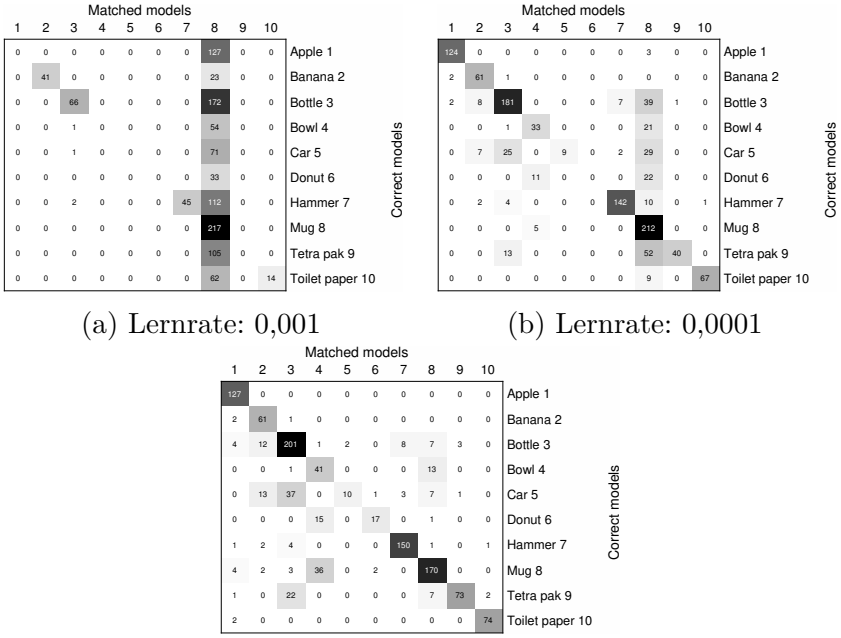


(a) Lernrate: 0,001 (b) Lernrate: 0,0001



(c) Lernrate: 0,00001

Abbildung 3.2.2.5: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Tasse bei einer Trainingszahl von 10.



(a) Lernrate: 0,001 (b) Lernrate: 0,0001

(c) Lernrate: 0,00001

Abbildung 3.2.2.6: Confusion Matrices des verkleinerten Testsets mit originalem Hintergrund. Bei Training auf einem Bild einer Tasse bei einer Trainingszahl von 100.

3.3 Iterative Trainingsprozesse

Dieser Abschnitt setzt sich mit den Ergebnissen des iterativen Algorithmus auseinander. Es wurden unterschiedliche Experimente und Evaluierungen durchgeführt, z. B. mit/ohne Bias Kontrolle, Augmentierung zu unterschiedlichen Zeitpunkten beim iterativen Trainingsprozess oder ob auf der Test- oder Trainingsdatenbank evaluiert wurde. Zu jeder einzelnen Untersuchung und auch nach dem Abschluss jedes iterativen Trainings wurden die fünf in Abschnitt 3.1.1 vorgestellten Matrizen gespeichert, sowie die Accuracy berechnet. Bei allen Evaluierungen wurde zum einen durch die Hintergrundfarbe der Bilder unterschieden, zum anderen wurde immer unterschieden ob iteratives Training bei einer Konfidenz unter 80 oder 90% angewendet wurde.

3.3.1 Keine Augmentierung und ohne *self-check*

Bei den hier präsentierten Ergebnissen wurde keinerlei Augmentierung im Rahmen der Klassifizierung oder des iterativen Trainings verwendet. Dies mag auf den ersten Blick durchaus korrekt erscheinen, muss allerdings in diesem Rahmen zu nicht ganz korrekten Ergebnissen führen, da der Algorithmus so auf Daten, welche im Trainingsprozess verwendet wurden, evaluiert wird. In den Tabellen 3.2, 3.3 und 3.4 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierungen im Rahmen der Trainingsprozesse

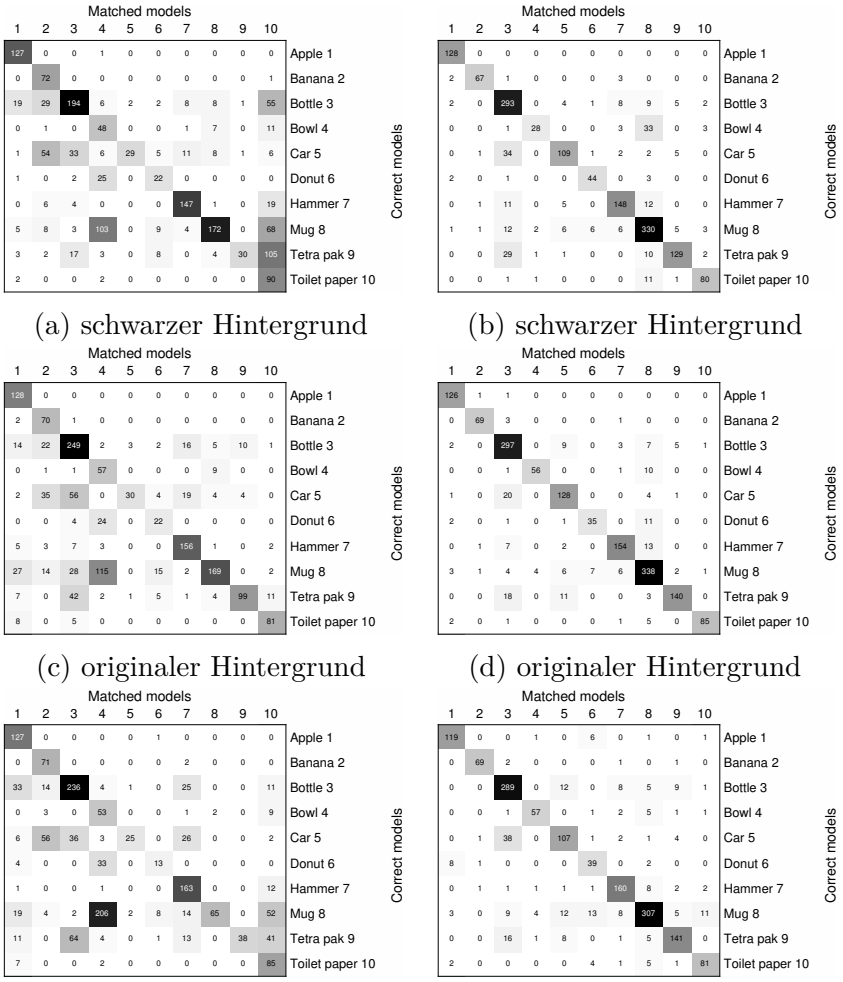
Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	57,75%	84,11%	87,77%
original	65,81%	88,58%	90,50%
weiß	54,34%	84,92%	88,95%

Tabelle 3.2: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

In Abbildung 3.3.1.1 sind rechts die CMs nach dem iterativen Lernvorgang, sofern die Konfidenz unter 80% liegt, und links die CMs vor dem Lernvorgang dargestellt. Es ist sowohl in Accuracy als auch in dem CMs bei allen Hintergründen eine Verbesserung der Klassifizierung zu sehen. Die Accuracy konnte um ca. $20 - 30\%$ erhöht werden. Allerdings konnte der grundlegende Bias(Flasche) hier nicht komplett ausgeglichen werden. Weiters tritt beim schwarzen Hintergrund

erneut die schon weiter oben beschriebene Verwechslung von Schüssel und Tasse auf.

Abbildung 3.3.1.2 zeigt die CMs für den iterativen Lernprozess sofern die Konfidenz unter 90% beträgt. Die CMs sowie Accuracy konnten gegenüber dem iterativen Training unter 80% Konfidenz noch um ca. $2-4\%$, da hier der grundlegende Bias besser behoben werden konnte.

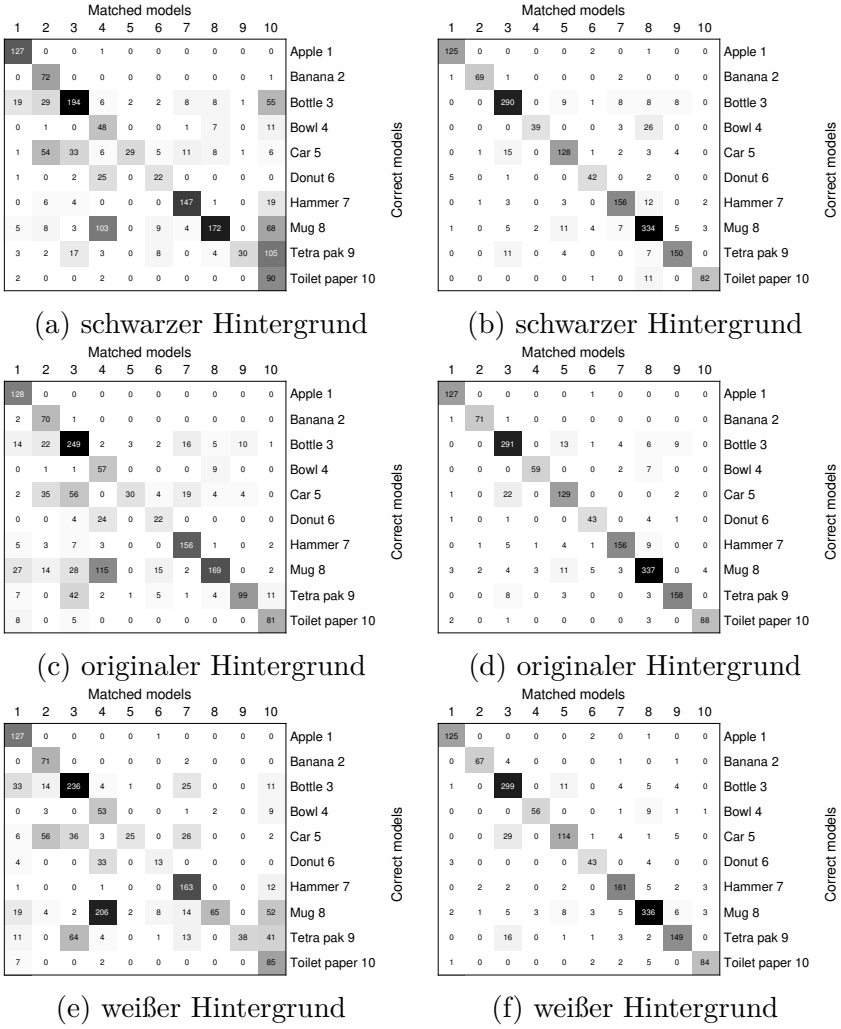


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.1.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.1.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und ohne *self-check*.

Da die Klassifizierer hier teilweise auf unterschiedlichen Daten trainiert wurden und so auch unterschiedliche Ergebnisse zustande kommen können, wurden um einen besseren Vergleich zu ermöglichen beide nochmals auf Test- und Trainingsdatenbank evaluiert. Dies ermöglicht auch bessere Vergleiche zu vor dem iterativen Training, da sich der Klassifizierer hier während des Klassifizierungsvorganges ändert.

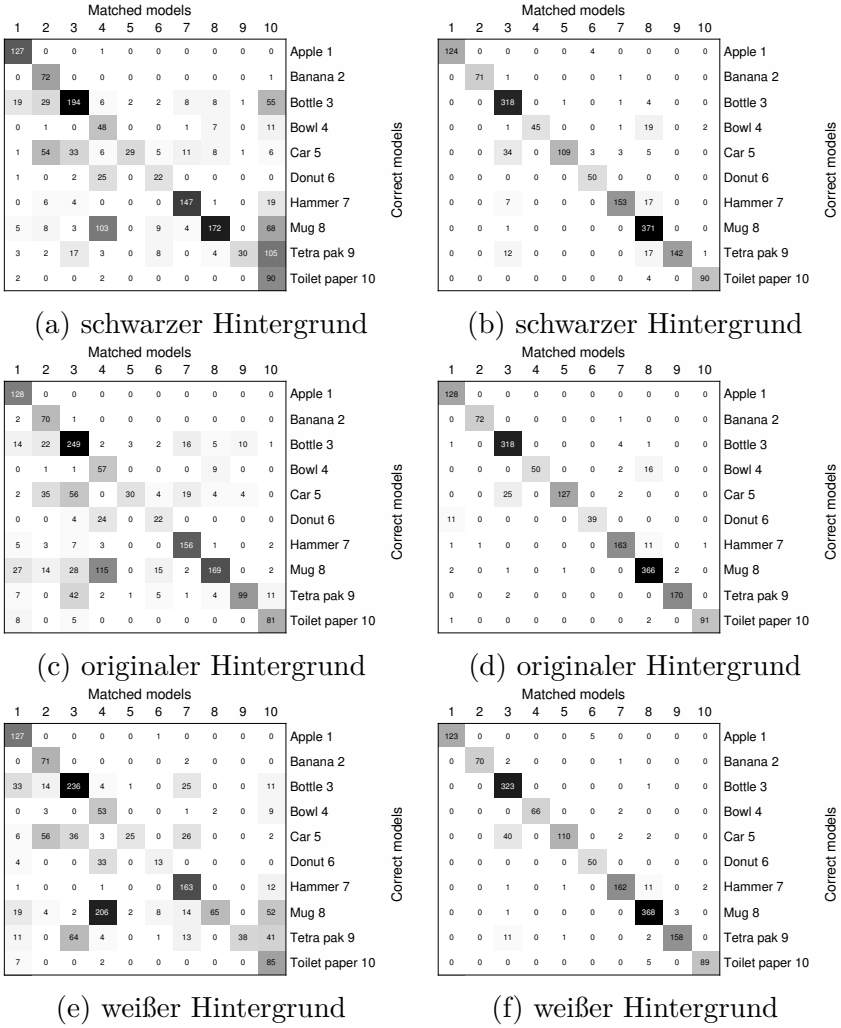
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	57,75%	91,37%	93,42%
original	65,81%	94,54%	94,78%
weiß	54,34%	94,23%	89,95%

Tabelle 3.3: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Zunächst wird der Klassifizierer beschrieben, welcher iterativ bei einer Konfidenz unter 80% trainiert wurde: Die CMs(Abbildung 3.3.1.3) sowie die Accuracy weisen auch hier eine Verbesserung von ca. $30 - 40\%$ auf. Bei originalem Hintergrund konnte der grundlegende Bias weitgehend behoben werden. Allerdings wurden bei schwarzem sowie originalem Hintergrund Schüsseln als Tassen klassifiziert.

Bei einem iterativen Training, sofern die Konfidenz unter 90% liegt weisen die CMs(Abbildung 3.3.1.4) bei jeglichem Hintergrund einen leichten Bias auf, bei schwarzem Hintergrund bei der Tasse, bei originalem Hintergrund bei der Flasche und bei weißem Hintergrund bei dem Hammer. Der Grund für diese Verzerrungen kann in der Überrepräsentation der jeweiligen Klasse bei den iterativen Trainingsprozessen liegen. Die Accuracy konnte hier gegenüber dem iterativen Training bei einer Konfidenz unter 80% bei schwarzem und originalem Hintergrund um ca. 2 bzw. 0,2% verbessert werden bei weißem Hintergrund nahm die Accuracy um ca. 5% ab.

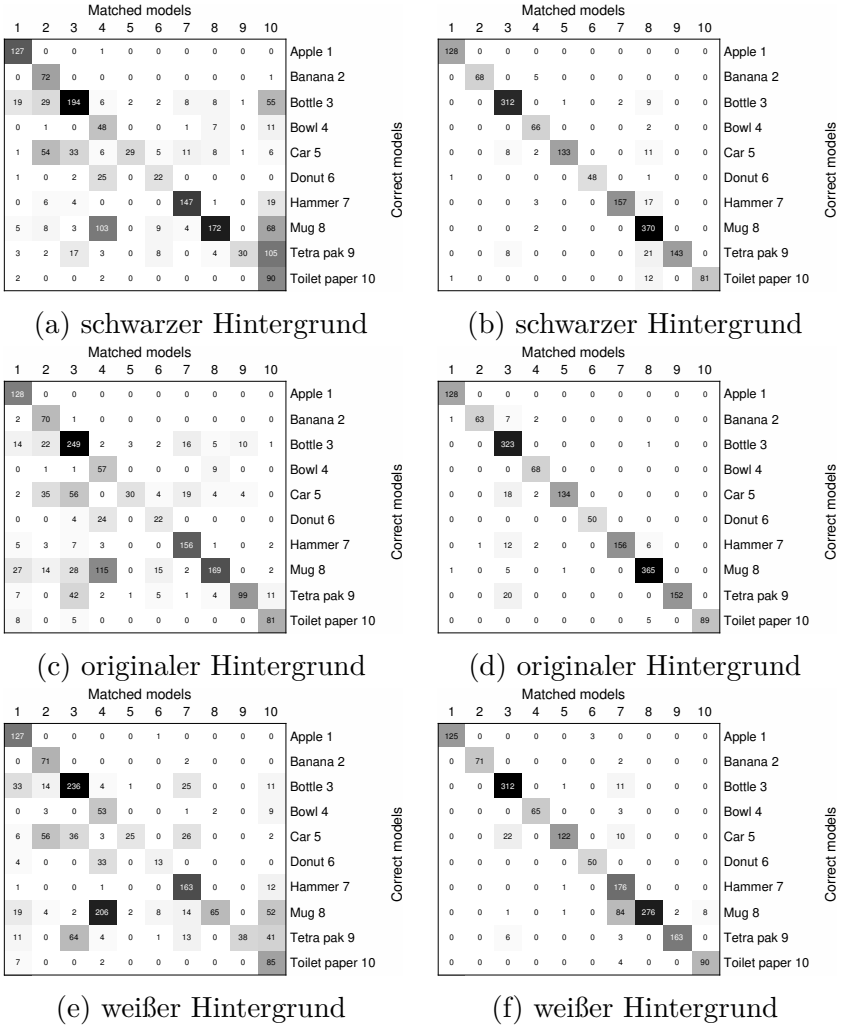


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.1.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.1.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und ohne *self-check*.

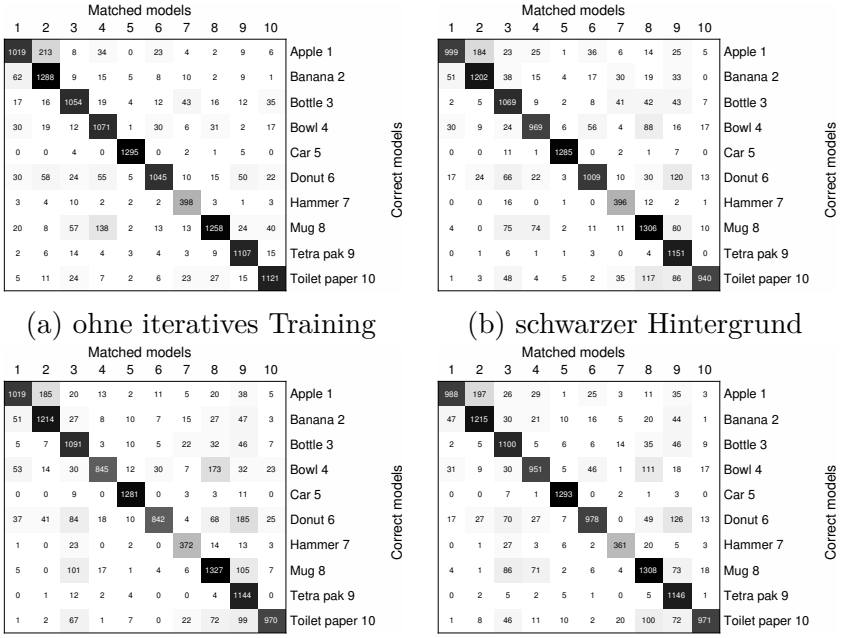
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	87,31%	84,61%	82,47%
original	87,31%	82,80%	80,44%
weiß	87,31%	84,48%	82,51%

Tabelle 3.4: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Bei iterativem Training, bei unter 80% Konfidenz zeigt sich sowohl in den CMs(Abbildung 3.3.1.5) als auch in der jeweiligen Accuracy ist eine leichte Verschlechterung um ca. $3-5\%$ gegenüber den Ergebnissen ohne iterativem Training. Dies ist auf die Tatsache zurückzuführen, dass das CNN nun mehr auf Daten der Testdatenbank angepasst wurde und so die Trainingsdatenbank weniger gut unterscheiden kann.

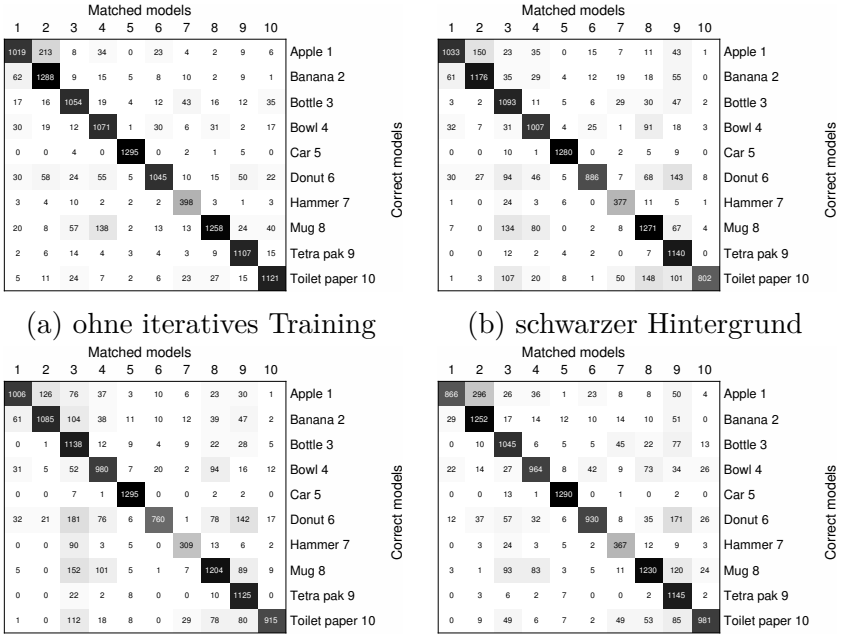
Bei Training bei unter 90% Konfidenz zeigt sich der oben beschriebene Effekt sowohl in den CMs(Abbildung 3.3.1.6) als auch der jeweiligen Accuracy(ca. $5-7\%$ geringer) deutlicher. Da hier auf mehr Daten der Testdatenbank trainiert wurde.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.1.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und ohne *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.1.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und ohne *self-check*.

3.3.2 Augmentierung beim iterativen Trainingsprozess und ohne *self-check*

Bei den hier präsentierten Ergebnissen wurden die Daten bei der Klassifizierung nicht verändert, bei dem iterativen Lernen schon. Dies ermöglicht das erneute Evaluieren des Algorithmus auf den Daten. In den Tabellen 3.5, 3.6 und 3.7 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierungen im Rahmen der Trainingsprozesse

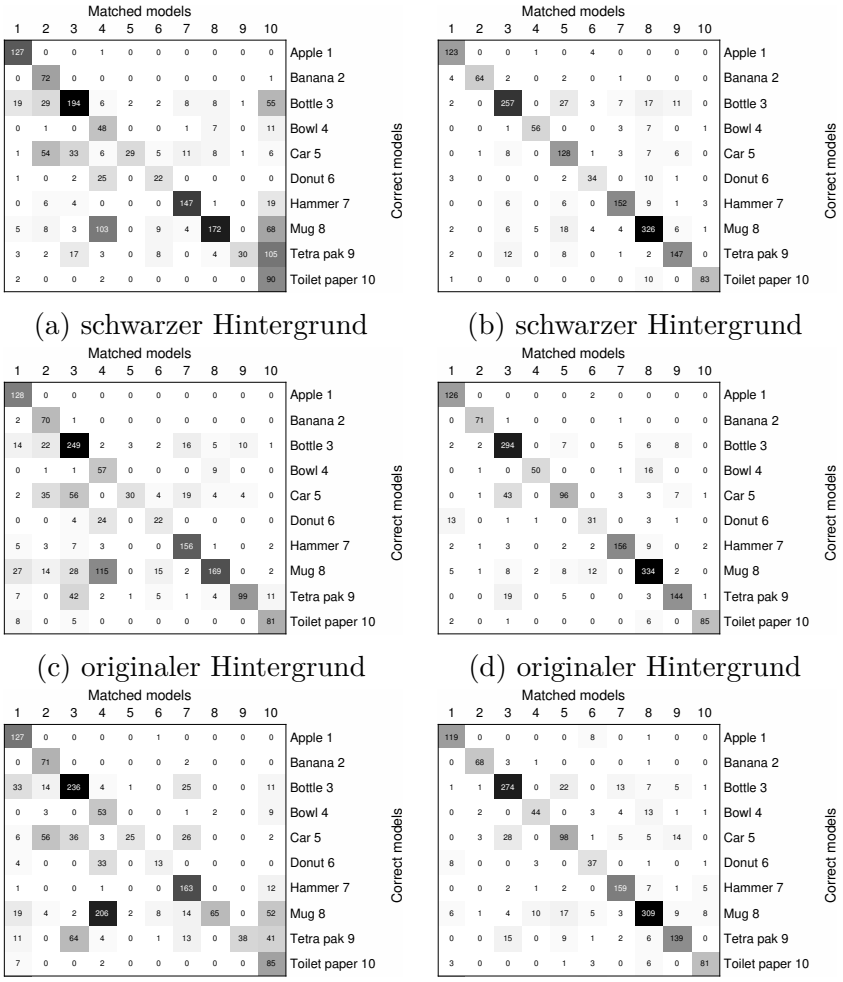
Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	57,75%	84,98%	87,22%
original	65,81%	86,04%	89,70%
weiß	54,34%	82,38%	87,40%

Tabelle 3.5: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.2.1 zeigt die CMs nach den iterativen Lernprozessen bei einer Konfidenz unter 80%. Die CMs und die Accuracy ist hier immer noch deutlich besser als ohne iterativem Lernen(ca. $20 - 30\%$). Gegenüber dem Training ohne Augmentierung haben sich die Ergebnisse allerdings bei weißem und originalem Hintergrund leicht verschlechtert(ca. 2%). Der grundlegende Bias(Flasche) ist auch hier bei originalem und weißem Hintergrund vorhanden, zusätzlich weisen die Klassifizierer mit schwarzem und weißem Hintergrund auch andere Bias auf. Beim 'schwarzen' Klassifizierer bei Objekt 5(Auto) und beim 'weißen' bei den Objekten 5(Auto), 8(Tasse) und 9(Tetra Pack).

Bei iterativem Training, bei einer Konfidenz unter 90% zeigen die CMs (Abbildung 3.3.2.2) und die Accuracy auch hier bessere Ergebnisse als ohne iteratives Training(ca. $20 - 30\%$) und ebenso teilweise nicht so gute Ergebnisse wie bei Training ohne Augmentierung(ca. $0{,}5 - 1\%$). Hier konnte wie auch schon bei Training ohne Augmentierung der grundlegende Bias bei schwarzem und originalem Hintergrund nahe zu komplett ausgebessert werden. Bei weißem Hintergrund konnte der Bias hier deutlich abgeschwächt werden, was die besseren Ergebnisse erklärt. Allerdings wurde bei allen Hintergrundfarben ein leichter Bias durch Objekt 5(Auto) eintrainiert, was die etwas schlechteren Ergebnisse erklärt.

Auch hier wurde wie bei anderen durchgeführten Lernprozessen anschließend an den Trainingsprozess die Test- und Trainingsdatenbank erneut klassifiziert um hier Daten zu erhalten, während sich der jeweilige Klassifizierer nicht verändert.

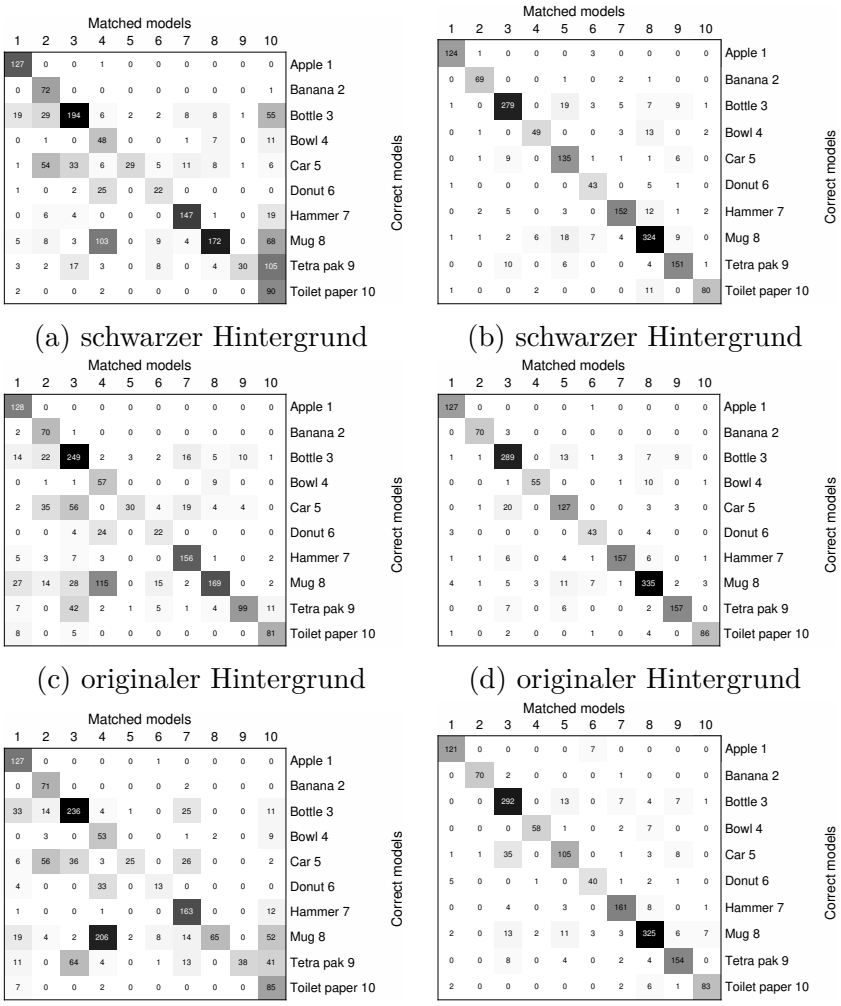


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.2.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.2.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.

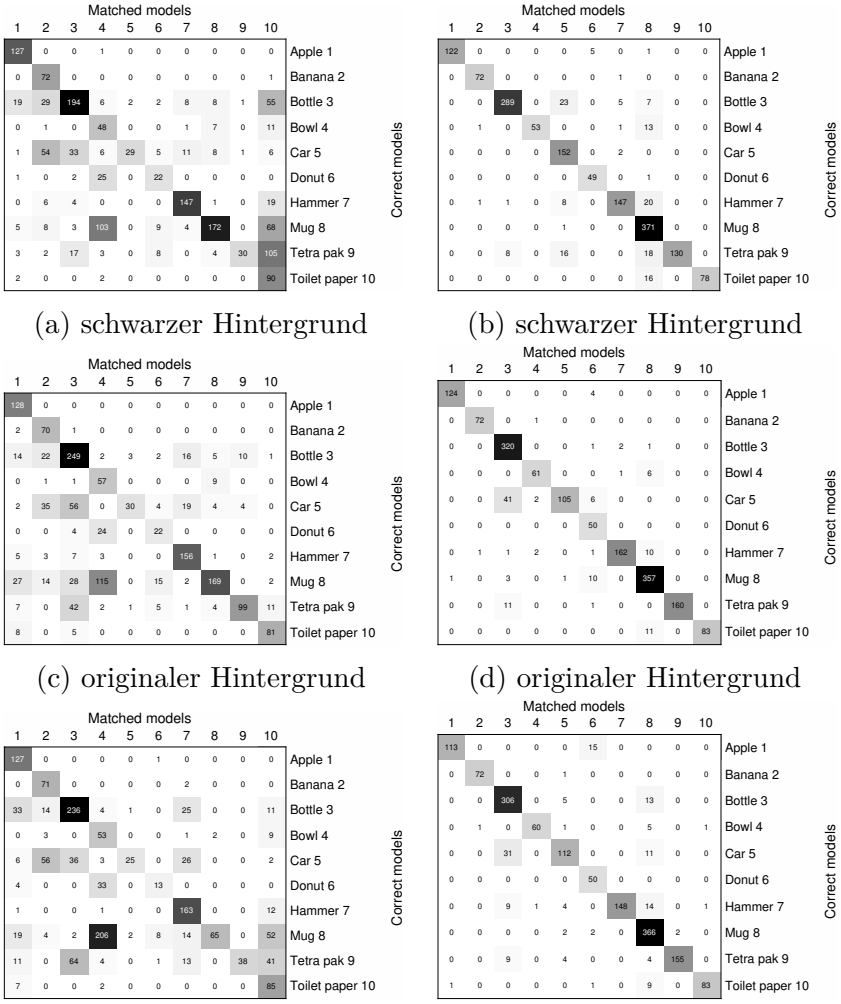
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	90,75%	95,84%
original	65,81%	92,69%	94,66%
weiß	54,34%	90,88%	86,97%

Tabelle 3.6: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.2.3 zeigt die CMs jener Klassifizierer, welche zuvor iterativ bei einer Konfidenz unter 80% trainiert wurden. Die Ergebnisse decken sich mit jenen, welche während der Trainingsprozesse entstanden sind: Im Vergleich zu keinem iterativen Training ist eine deutliche Verbesserung zu erkennen(ca. $27 - 36\%$). Im Vergleich zu iterativem Training ohne Augmentierung ist eine leichte Verschlechterung zu erkennen(ca. $1 - 3\%$). Die Bias-Fehler konnten bei allen Hintergründen nahezu behoben werden, der 'schwarze' Klassifizierer weist einen leichte Bias bei Objek 5(Auto) und bei Objekt 8(Tasse), der 'originale' den grundlegenden Bias bei Objekt 3(Schüssel) und der 'weiße' eine bei Objekt 3(Schüssel) und bei Objekt 8(Tasse) auf.

Sofern der Klassifizierer bei einer Konfidenz unter 90% iterativ trainiert wurden sind die Ergebnisse(siehe Abbildung 3.3.2.4) dieser Klassifizierung besonders interessant, da alle 3 Klassifizierer unterschiedliche Entwicklungen zeigen. Der Klassifizierer welcher mit schwarzem Hintergrund arbeitet zeigt hier eine deutliche Verbesserung sogar gegenüber jenem, welcher ohne Augmentierung trainiert wurde(ca. 2%). Die CM zeigt eine klar ausgeprägt Diagonale und sonst nahezu keine anderen signifikant hohen Werte. Der 'originale' Klassifizerer zeigt eine deutliche Verbesserung gegenüber jenem, welcher nicht iterativ trainiert wurde,(ca. 29%) allerdings keine deutliche Veränderung bezüglich Accuracy gegenüber jenem Klassifizierer, welcher ohne Augmentierung trainiert wurde. Lediglich der Bias wurde besser korrigiert, allerdings wurde Objekt 8(Tasse) ein wenig verlernt, was die geringe Änderung in Accuracy erklärt. Beim 'weißen' Klassifizierer trat eine Verbessrung gegenüber keinem iterativen Training auf(ca. 34%), allerdings eine Verschlechterung gegenüber keiner Augmentierung(ca. 3%). Es wurde sichtlich ein Bias bei Objekt 5(Auto) eintrainiert, was die schlechten Ergebnisse erklärt.

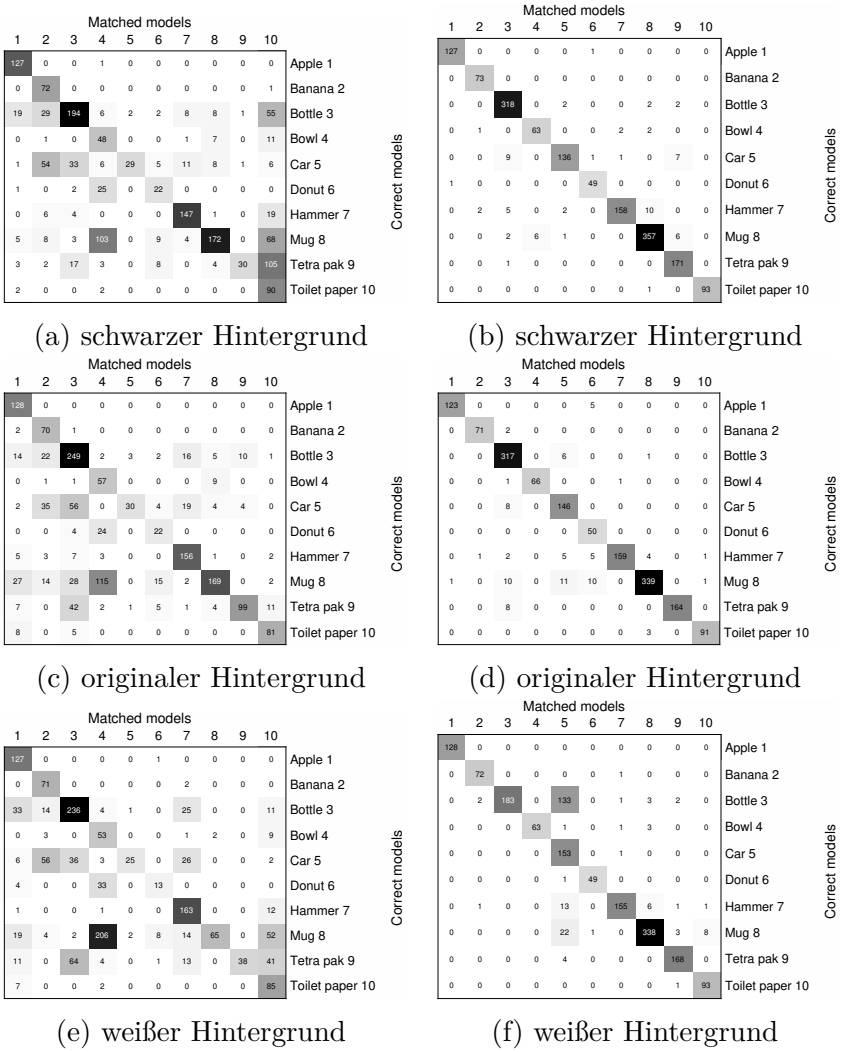


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.2.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.2.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.

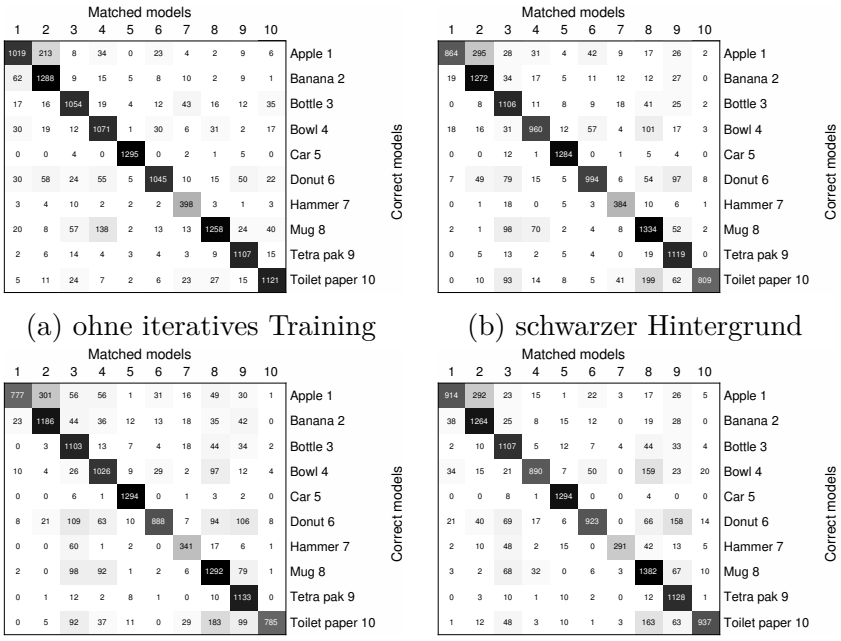
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	87,31%	82,97%	79,63%
original	87,31%	80,50%	77,54%
weiß	87,31%	83,00%	80,66%

Tabelle 3.7: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.2.5 zeigt die CMs der Trainingsdatenbanken sofern der jeweilige Klassifizierer iterativ bei einer Konfidenz unter 80% trainiert wurde. Die Ergebnisse decken sich mit den bisherigen: Es tritt eine leichte Verschlechterung gegenüber den Klassifzierungen ohne iterativem Training(ca. $5-7\%$) und auch gegenüber iterativem Training ohne Augmentierung auf(ca. 2%). Die Gründe hierfür wurden weiter oben schon erläutert. In den CMs sind leichte Bias mehrerer Objekte bei allen drei Hintergründen ersichtlich: bei schwarzem Hintergrund Objekte 3(Flasche), 8(Tasse) und 9(Tetra Pack), bei originalem Hintergrund Objekte 3(Flasche), 4 (Schüssel), 8(Tasse) und 9(Tetra Pack) und bei weißem Hintergrund ebenfalls Objekte 3(Flasche), 8(Tasse) und 9(Tetra Pack). Die Tatsache, dass es sich bei allen Hintergründen um dieselben Objekte handelt, bei welchen ein Bias auftritt, lässt darauf schließen, dass diese Objekte allgemein schwer zu erkennen sind.

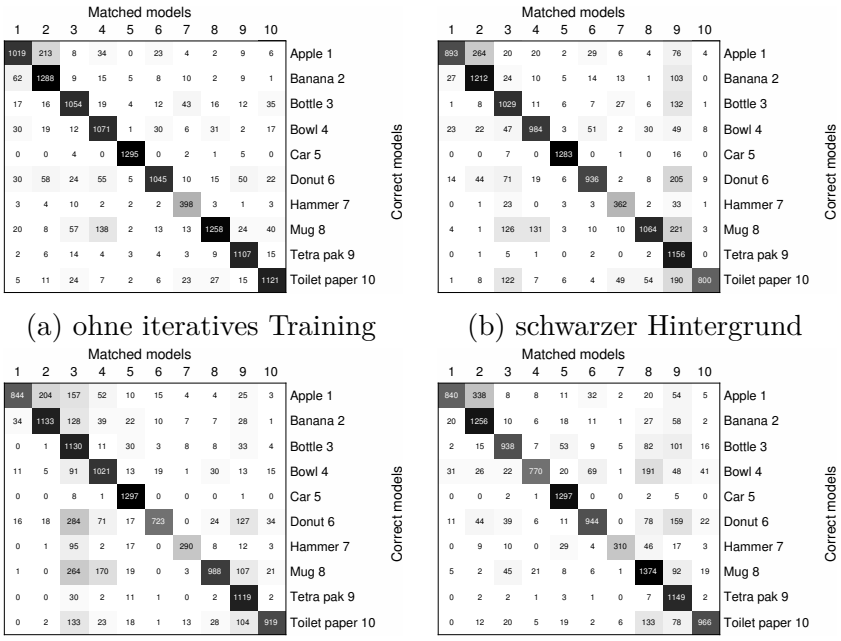
Die CMs für die Trainingsprozesse der Klassifizierer bei einer Konfidenz unter 90% sind in Abbildung 3.3.2.6 zu finden. Die Ergebnisse aus diesen Klassifizierungen decken sich auch hier mit den oben schon gewonnenen Erkenntnissen, dass Training bei einer Konfidenz unter 90% die Klassifizierung auf der Trainingsdatenbank verschlechtert(ca. $7-10\%$). Allerdings zeigt sich hier ein anderer Trend bezüglich der CMs als bei iterativem Training unter 80% Konfidenz. Hier zeigt der 'schwarze' Klassifizierer einen deutlicheren Bias bei Objekt 9(Tetra Pack), der 'originale' einen deutlicheren bei Objekt 3(Flasche) und der 'weiße' einen Bias bei den Objekten 8(Tasse) und 9(Tetra Pack). Dies zeigt, dass die Klassifizierer die Objekte scheinbar doch unterschiedlich gut lernen und nicht wie zuvor angenommen gleich gut.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.2.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.2.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und ohne *self-check*.

3.3.3 Komplette Augmentierung und ohne *self-check*

Bei den hier präsentierten Ergebnissen wurden die Daten bei der Klassifizierung im Rahmen der Trainingsprozesse zufällig augmentiert und auf diesen Daten trainiert. Dies ermöglicht das erneute Evaluieren des Algorithmus auf den Daten. In den Tabellen 3.8, 3.9 und 3.10 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierung im Rahmen der Trainingsprozesse

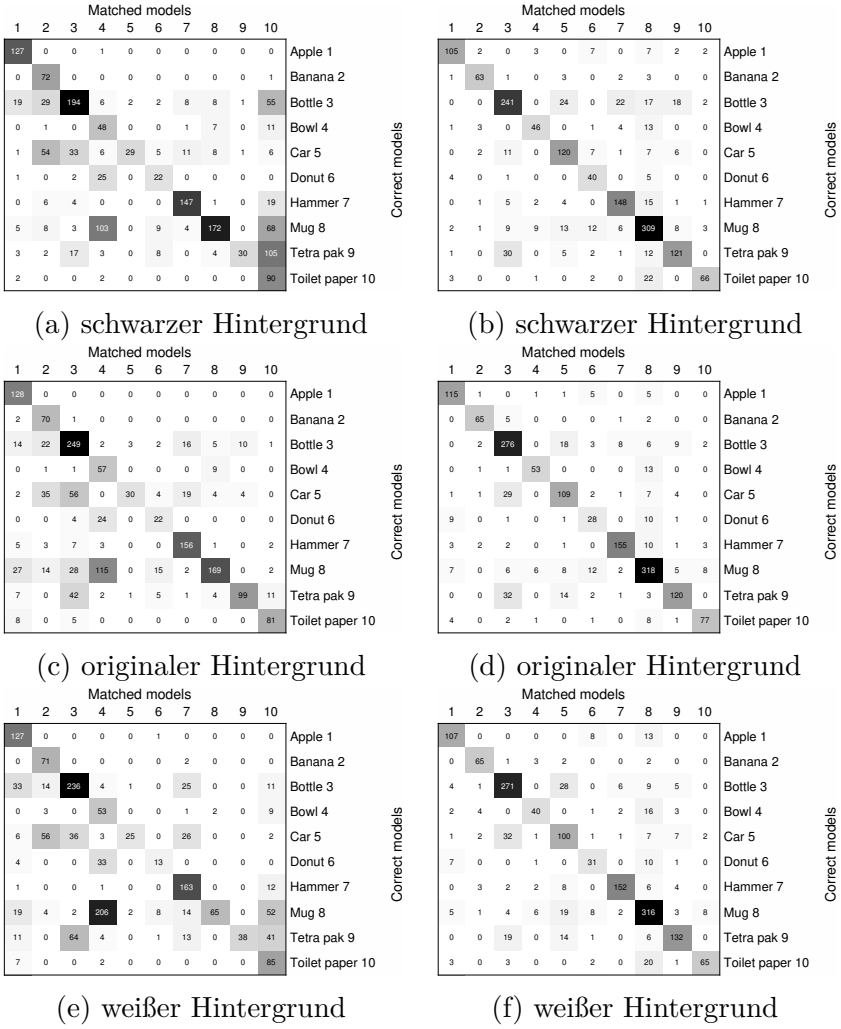
Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	57,75%	78,10%	79,21%
original	65,81%	81,63%	81,76%
weiß	54,34%	79,34%	80,83%

Tabelle 3.8: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.3.1 zeigt die CMs bei iterativem Training bei einer Konfidenz unter 80%. Bei diesem Trainingsprozess findet zwar noch eine Verbesserung, sowohl der Accuracy(ca. $16 - 25\%$) als auch der jeweiligen CM statt, allerdings ist die Verbesserung im Vergleich zu den anderen beiden Trainingsprozessen ohne Bias Überprüfung am geringsten. Die CM zeigen immer noch eine deutliche Verbesserung, jedoch werden bei schwarzem und weißem Hintergrund nicht mehr alle Objekte korrekt klassifiziert, z. B. Objekt 3(Flasche). Alle drei Klassifizierer zeigen neben dem grundlegend Bias(Objekt 3) auch einen leichten Bias bei Objekt 8(Tasse), was darauf schließen lässt, dass durch die zufällige Augmentierung der Daten diese Klasse zu oft trainiert werden muss.

Bei iterativem Training bei einer Konfidenz von unter 90% sind die CMs in Abbildung 3.3.3.2 dargestellt. Sowohl CMs, als auch die Accuracy zeigen ein leicht besseres Ergebnis(ca. 1%) im Vergleich zum Training bei einer Konfidenz unter 80%, doch der oben beschriebene Fall, dass Objekte teilweise nicht mehr korrekt klassifiziert werden, tritt auch hier auf. Im Vergleich zu den anderen beiden Trainingsprozessen ohne Augmentierung kann hier auch der grundlegende Bias (Objekt 3) nicht mehr so gut korrigiert werden.

Auch hier wurde wie bei anderen durchgeführten Lernprozessen anschließend an den Trainingsprozess die Test- und Trainingsdatenbank erneut klassifiziert um hier Daten zu erhalten, während sich der jeweilige Klassifizierer nicht verändert und die Daten bei der Klassifizierung nicht augmentiert sind.

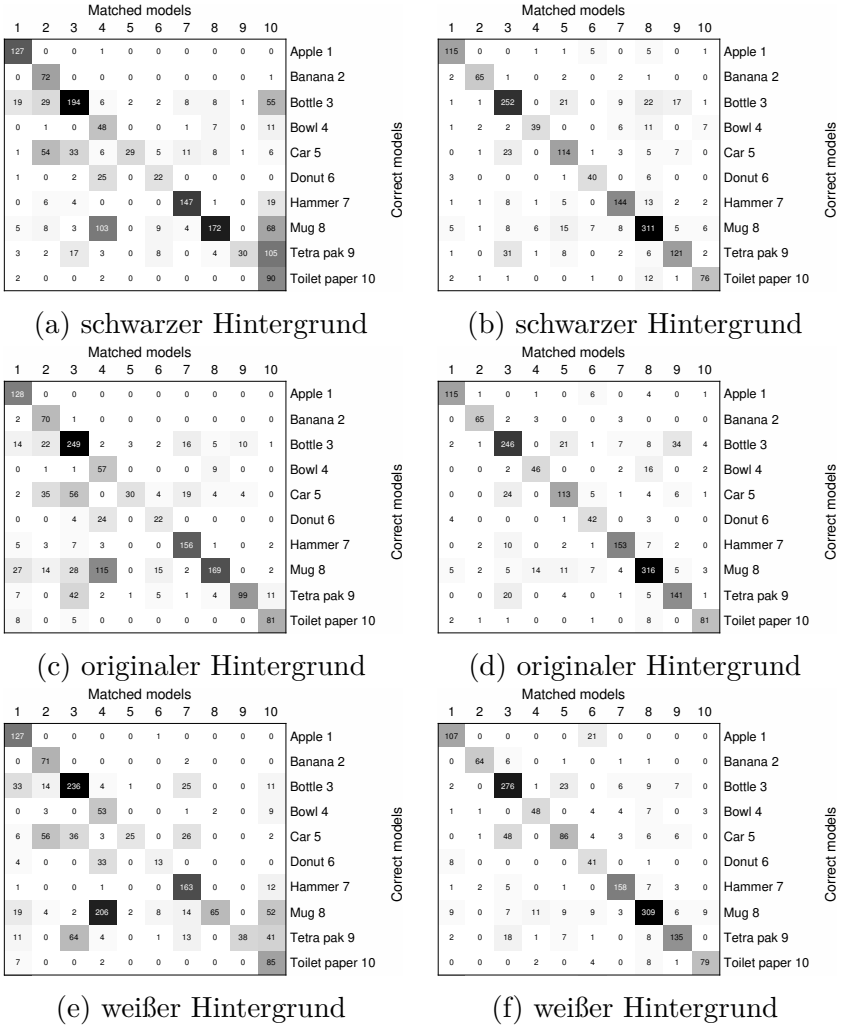


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.3.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.3.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.

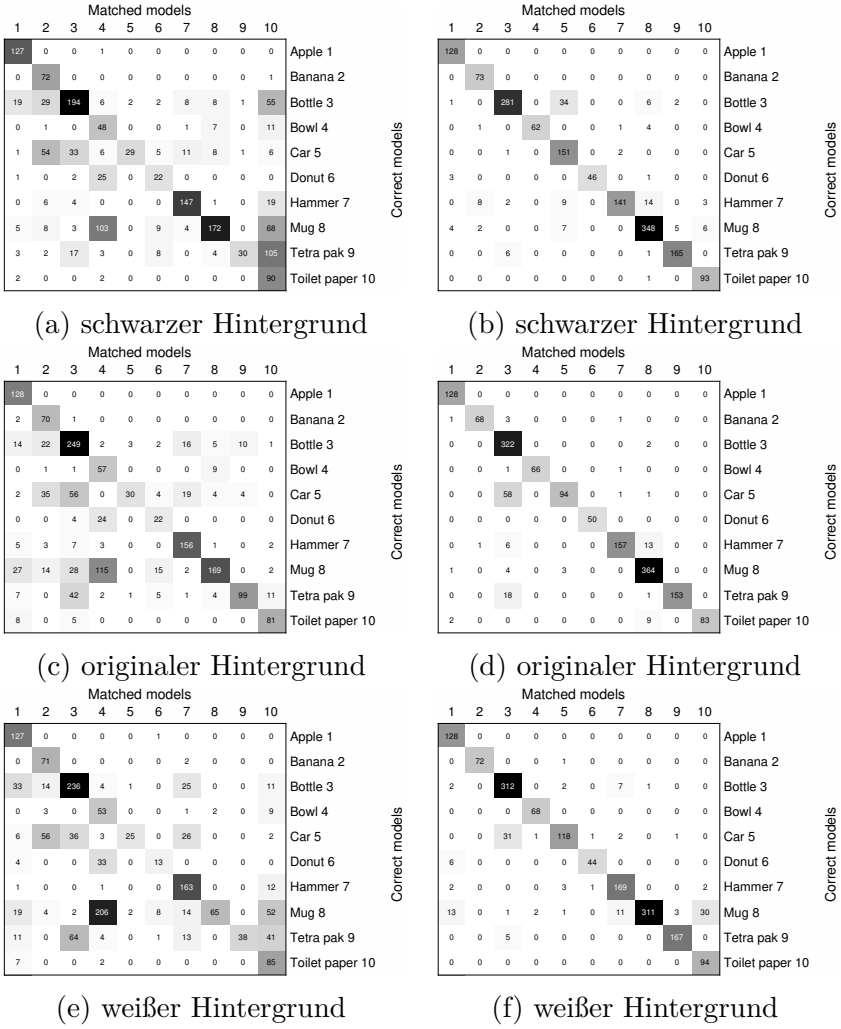
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	92,30%	91,99%
original	65,81%	92,12%	94,04%
weiß	54,34%	91,99%	90,44%

Tabelle 3.9: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.3.3 zeigt die CMs für iteratives Training bei einer Konfidenz unter 80%. Auch in diesem Fall ist eine deutliche Verbesserung gegenüber keinem iterative Training bei CM und Accuracy sichtbar(ca. $27 - 37\%$). Der 'schwarze' Klassifizierer zeigt hier sogar bessere Ergebnisse, als ohne Augmentierung erzielen werden, wogegen die anderen beiden schlechtere Ergebnisse liefern. Gegenüber dem Trainieren auf augmentierten Bildern konnte der 'schwarze' Klassifizierer sich um ca. 1% verbessern, der 'originale' und 'weiße' verschlechterten sich um ca. 0,5 bzw. 1%. Allgemein weißt der 'schwarze' Klassifizierer einen Bias bei Objekt 5 auf . Der 'originale' weist den grundlegenden Bias auf. Der 'weiße' Klassifizierer erkennt Objekt 8 nicht ganz korrekt. Die Gründe hierfür können in der Augmentierung liegen, z. B. Objekt 8 beim 'weißen', da auch Helligkeiten verändert werden. Es kann auch zu Überrepräsentationen beim Trainingsprozess kommen(Objekt 5 beim 'schwarzen oder Objekt 3 beim 'originalen').

Bei iterativem Training, bei einer Konfidenz unter 90% zeigen die CMs(siehe Abbildung 3.3.3.4) sowie die Accuracy eine Verbesserung gegenüber keinem iterativen Training(ca. $29 - 36\%$). Durch die frühere Augmentierung scheinen sich allerdings die Ergebnisse teilweise maßgeblich zu verändern. Der 'schwarze' Klassifizierer kann Objekt 8 nicht mehr korrekt klassifizieren, beim 'originalen' wird der grundlegende Bias nicht korrigiert und der 'weiße' hat einen Bias bei Objekt 5. Diese Veränderungen im Vergleich zum Training, bei dem Augmentierung lediglich beim Training angewandt wird lässt den Schluss zu, dass die Augmentierung bei der Klassifizierung den Trainingsprozess verschlechtert.

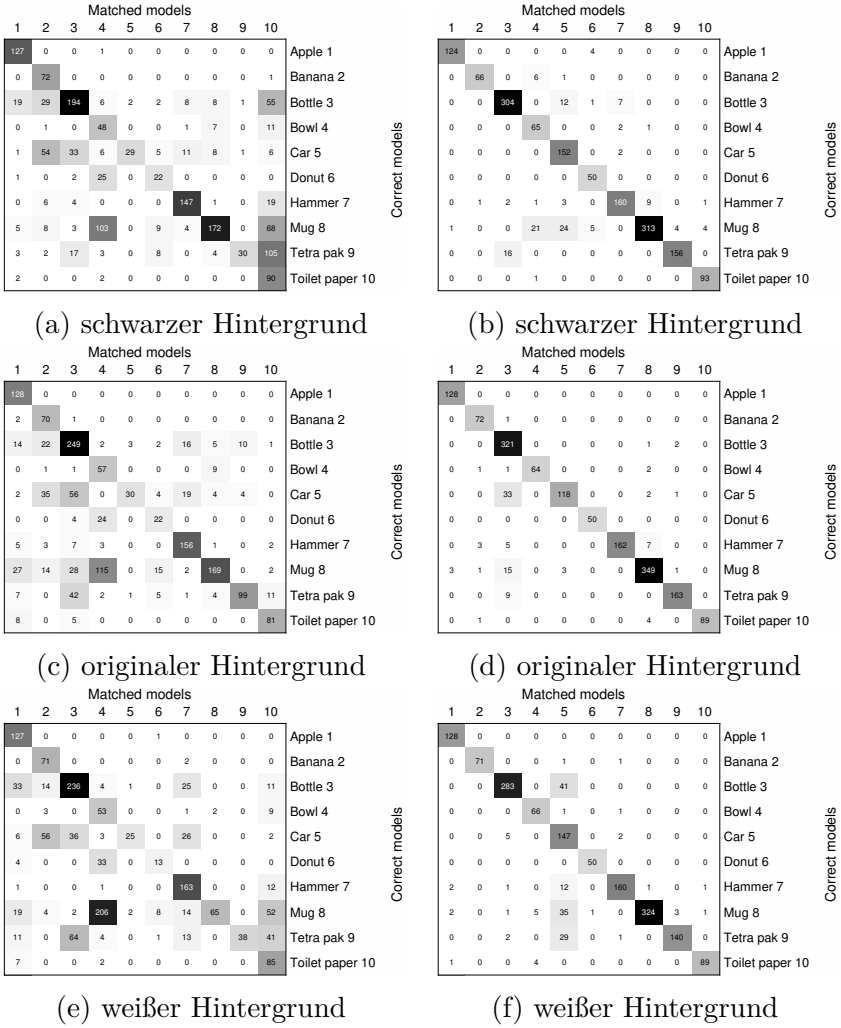


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.3.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.3.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.

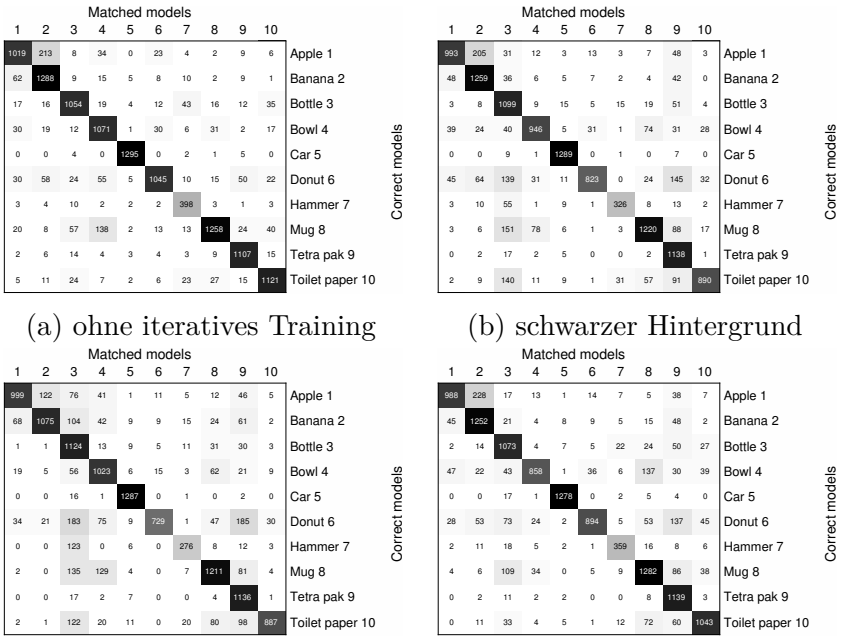
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	87,31%	81,80%	79,28%
original	87,31%	79,86%	75,95%
weiß	87,31%	83,30%	79,76%

Tabelle 3.10: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.3.5 zeigt die CMs der Trainingsdatenbanken, bei zuvor durchgeführtem iterativen Training, bei einer Konfidenz unter 80%. Hier zeigen sich erneut ähnliche Ergebnisse wie schon bei den zuvor durchgeführten Evaluierungen auf der Trainingsdatenbank. Der 'weiße' Klassifizierer zeigt bessere Ergebnisse als die beiden anderen, sowohl in der CM(Bias ist geringer) als auch in der geringeren Verschlechterung der Accuracy(ca. 4%). Dieser Umstand deckt sich mit der Tatsache, dass dieser Klassifizierer auf dem Testset keine so guten Ergebnisse liefert, da er scheinbar auf iteratives Training nicht so gut anspricht. Beim 'schwarzen' und 'originalen' zeigt sich der stärkste Bias bei Objekt 3, wobei beide auch bei anderen Objekten leicht beeinflusst sind.

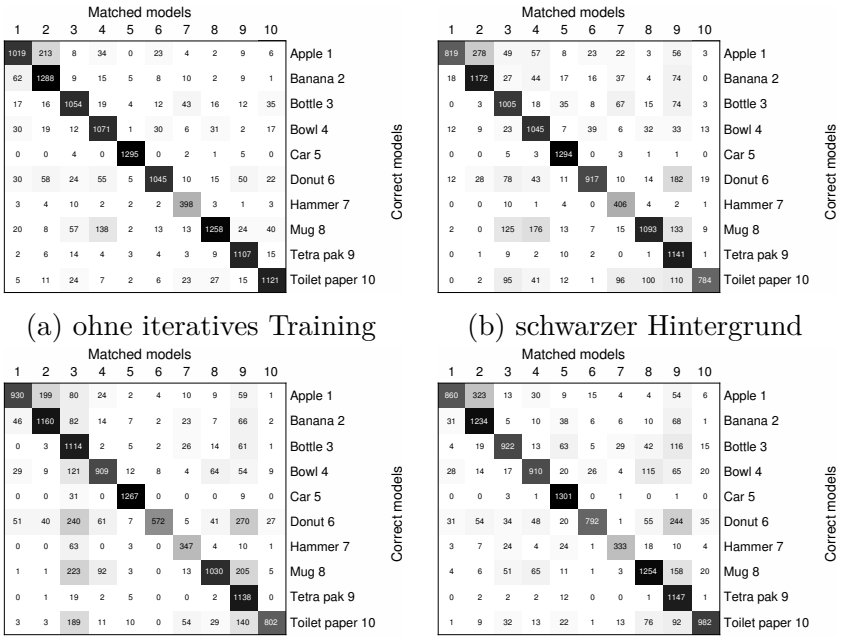
Bei iterativem Training bei einer Konfidenz unter 90% zeigt auch in diesem Fall der 'weiße' Klassifizierer erneut bessere Ergebnisse als die beiden anderen, wobei er hier auch einen Bias bei Objekt 9 aufweist. Interessant ist das Verhalten der beiden anderen Klassifizierer gegenüber dem Training bei einer Konfidenz unter 80%. Während beim 'schwarzen' lediglich die Accuracy um weniger als 2% abnimmt, wird der Bias beim 'originalen' bei Objekt 3 stärker während die Accuracy um ca. 4% abnimmt.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.3.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.3.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und ohne *self-check*.

3.3.4 Keine Augmentierung und mit *self-check*

Bei den hier präsentierten Ergebnissen wurde keinerlei Augmentierung im Rahmen der Klassifizierung oder des iterativen Trainings verwendet. Dies mag auf den ersten Blick durchaus korrekt erscheinen, muss allerdings in diesem Rahmen zu nicht ganz korrekten Ergebnissen führen, da der Algorithmus so auf Daten, welche im Trainingsprozess verwendet wurden, evaluiert wurde. Hier wurde allerdings erstmals *self-check* zur Bias-Überprüfung und -Korrektur angewandt. In den Tabellen 3.11, 3.12 und 3.13 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierungen im Rahmen der Trainingsprozesse

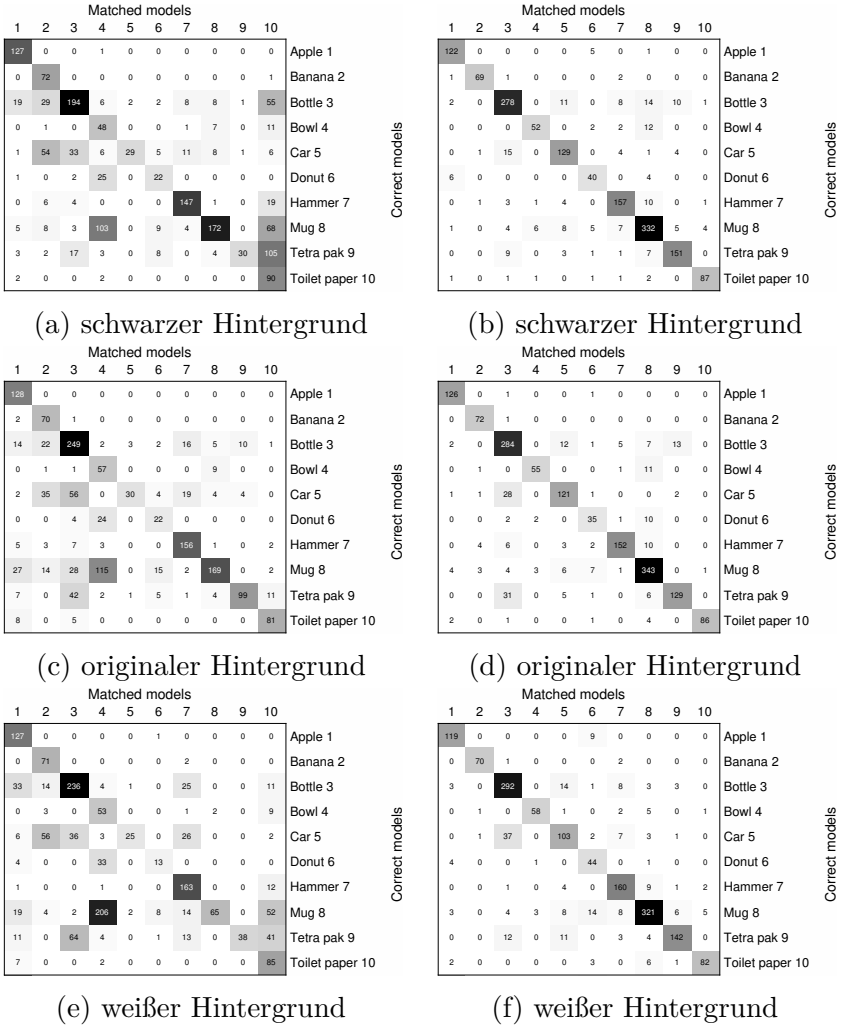
Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	87,90%	89,14%
original	65,81%	87,03%	90,13%
weiß	54,34%	86,29%	87,03%

Tabelle 3.11: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.4.1 zeigt die CMs bei iterativem Training bei einer Konfidenz unter 80%. Die CMs sowie die Accuracy der Klassifizierungen zeigen teilweise bessere Ergebnisse durch *self-check*. Der grundlegende Bias konnte beim 'schwarzen' und 'weißen' Klassifizierer besser ausgebessert werden, was in einer um ca. 3 bzw. 1% höheren Accuracy im Vergleich zu dem Trainingsprozess ohne *self-check* resultiert. Interessanterweise konnte der Bias beim 'originalen' nicht ausgebessert werden, was durch bessere Parameter behoben werden könnte. In diesem Fall ist auch die Accuracy um ca. 1,5.% gesunken.

Bei iterativem Training, bei einer Konfidenz unter 90% konnten in den CMs(siehe Abbildung 3.3.4.2) bei allen Klassifizieren etwa die selben Ergebnisse erzielt werden wie beim Trainingsprozess ohne *self-check*. Die Accuracy steigerte sich beim 'schwarzen' Klassifizierer um ca. 1% und nahm beim 'originalen' und 'weißen' um ca. 0,4 bzw. 2% ab.

Auch hier wurde wie bei anderen durchgeführten Lernprozessen anschließend an den Trainingsprozess die Test- und Trainingsdatenbank erneut klassifiziert um hier Daten zu erhalten, während sich der jeweilige Klassifizierer nicht verändert.

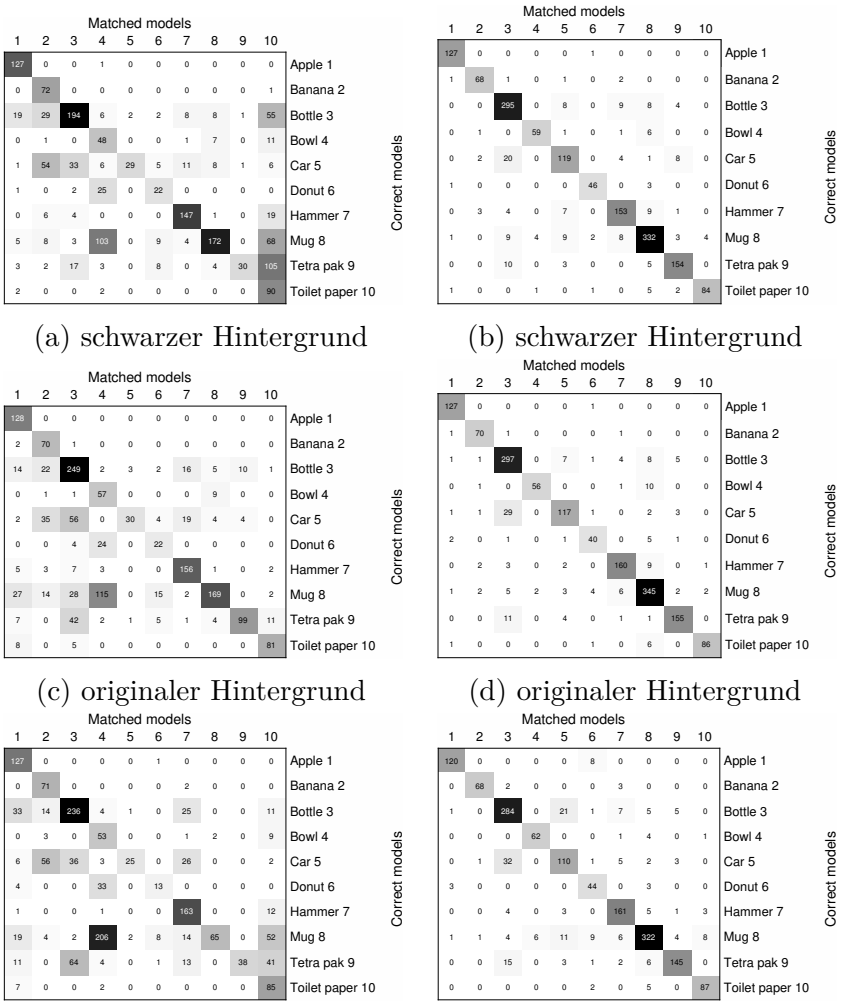


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.4.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und mit *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.4.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und mit *self-check*.

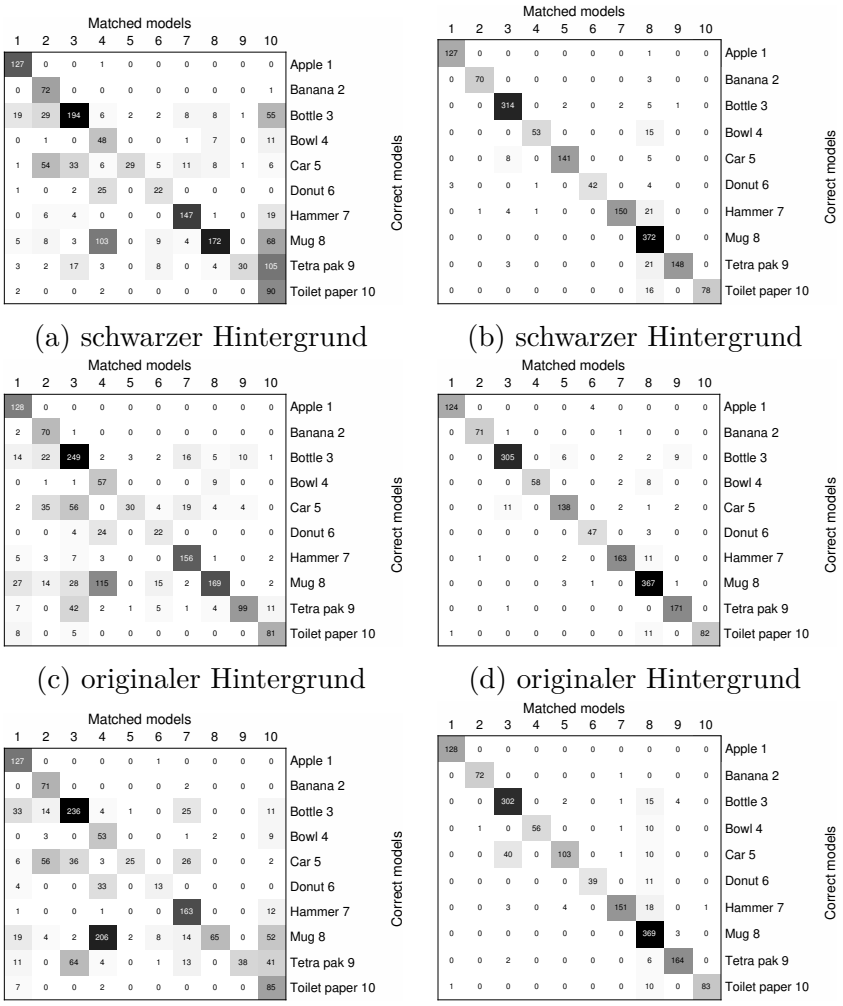
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	92,74%	95,78%
original	65,81%	94,66%	96,40%
weiß	54,34%	91,00%	91,43%

Tabelle 3.12: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.4.3 zeigt die CMs der Evaluierung der Testdatenbank bei iterativem Training bei einer Konfidenz unter 80%. In diesem Fall wurden die Ergebnisse durch *self-check* beim 'schwarzen' und beim 'originalen' Klassifizierer verbessert. Die Accuracy veränderte sich im Vergleich zum Training ohne *self-check* beim 'originalen' Klassifizierer nicht deutlich, während sie beim 'schwarzen' Klassifizierer um ca. 1,4% erhöht werden konnte. Beim 'weißen' tritt hier allerdings bei Objekt 8 ein Bias auf, was darauf hin deuten könnte, dass *self-check* bei weißem Hintergrund nicht so effektiv ist. Dies schlägt sich auch in einer, im Vergleich zum Training ohne *self-check* um 3% verringerten Accuracy nieder.

Bei iterativem Training, bei einer Konfidenz unter 90% ist gegenüber dem Training ohne *self-check* eine deutliche Verbesserung in den CMs(siehe Abbildung 3.3.4.4) sowie leichte höhere Accuracies(ca. 2%) zu erkennen. Der Bias konnte weitgehend abgeschwächt werden. Lediglich beim 'weißen' Klassifizierer kann die Verwechslung von Objekt 8 mit Objekt 7 nicht ausgebessert werden. Dies dürfte somit auf die weiße Hintergrundfarbe zurückzuführen sein.

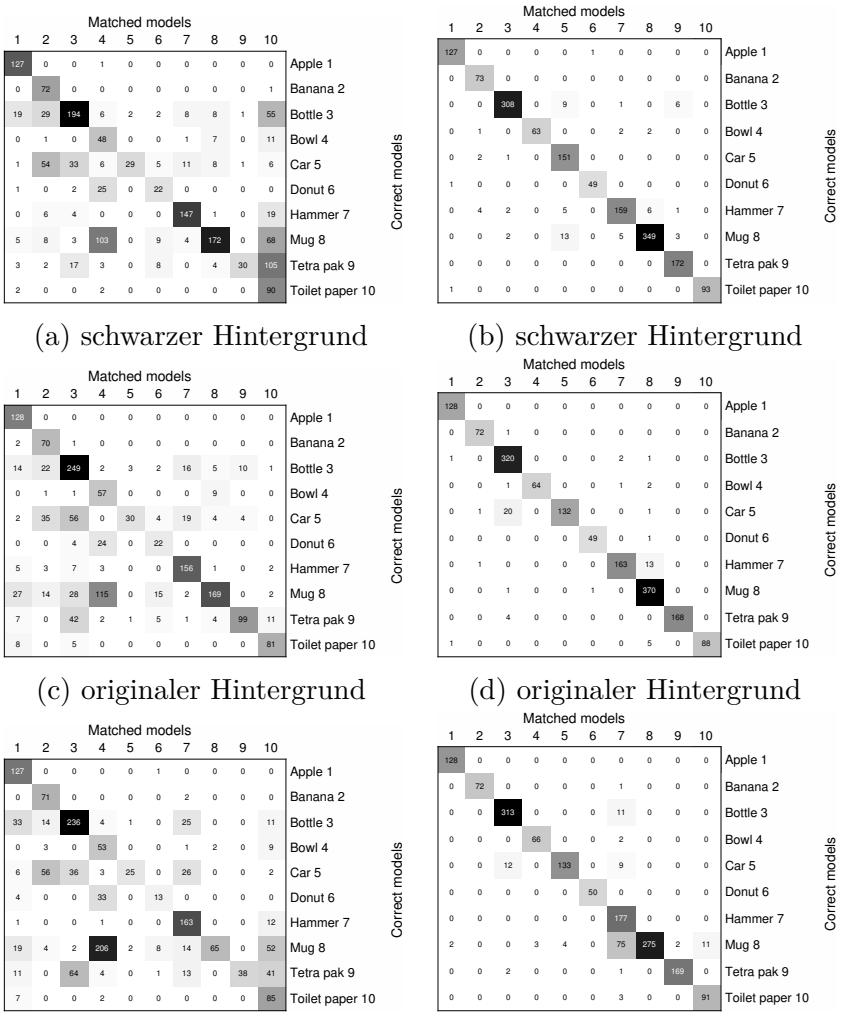


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.4.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und mit *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.4.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und mit *self-check*.

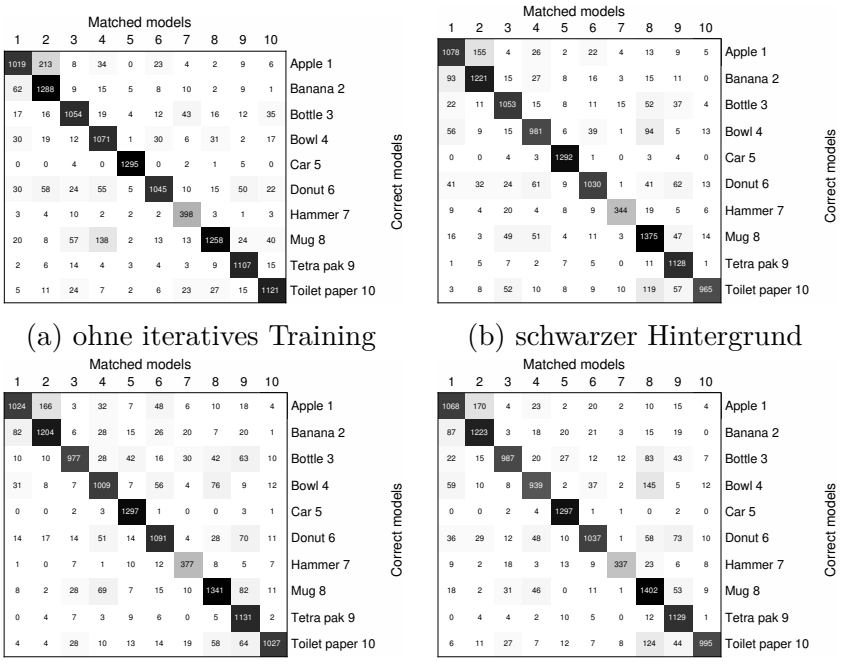
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	87,31%	85,76%	85,18%
original	87,31%	85,85%	85,17%
weiß	87,31%	85,33%	86,45%

Tabelle 3.13: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.4.5 zeigt die CMs der Trainingsdatenbank bei iterativem Training bei einer Konfidenz unter 80%. Hier wurde durch *self-check* die Accuracy bei allen Klassifizierern im Vergleich zu Training ohne *self-check* um ca. $1-3\%$ verbessert. Auch in den CMs ist sichtbar, dass der Bias leicht ausgebessert werden konnte.

Bei iterativem Training, bei einer Konfidenz unter 90% ist anders als bei dem Training ohne *self-check* keine deutliche Verschlechterung gegenüber dem Training bei einer Konfidenz unter 80% ersichtlich. Beim 'weißen' ist sogar eine leichte Verbesserung in der CM und Accuracy zu sehen. Die anderen beiden Klassifizierer weisen auch hier leicht abgeschwächte Bias auf.



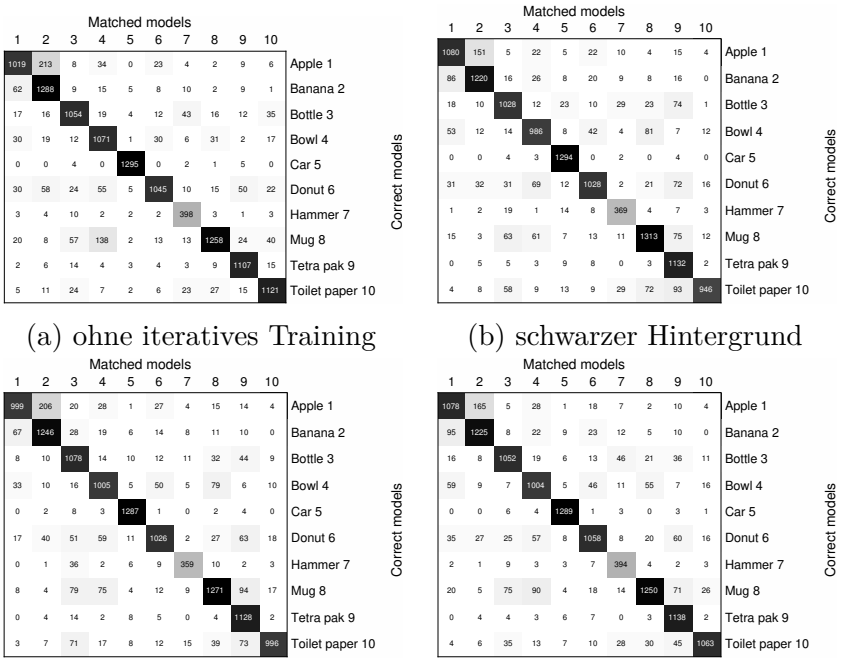
(a) ohne iteratives Training

(b) schwarzer Hintergrund

(c) originaler Hintergrund

(d) weißer Hintergrund

Abbildung 3.3.4.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz ohne Augmentierung und mit *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.4.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz ohne Augmentierung und mit *self-check*.

3.3.5 Augmentierung beim iterativen Trainingsprozess und mit *self-check*

Bei den hier präsentierten Ergebnissen wurden die Daten bei der Klassifizierung nicht verändert, bei dem iterativen Lernen schon. Dies ermöglicht das erneute Evaluieren des Algorithmus auf den Daten. Hier wurde ebenfalls *self-check* angewandt. In den Tabellen 3.14, 3.15 und 3.16 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierungen im Rahmen der Trainingsprozesse

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	86,35%	87,34%
original	65,81%	86,97%	88,70%
weiß	54,34%	84,30%	84,49%

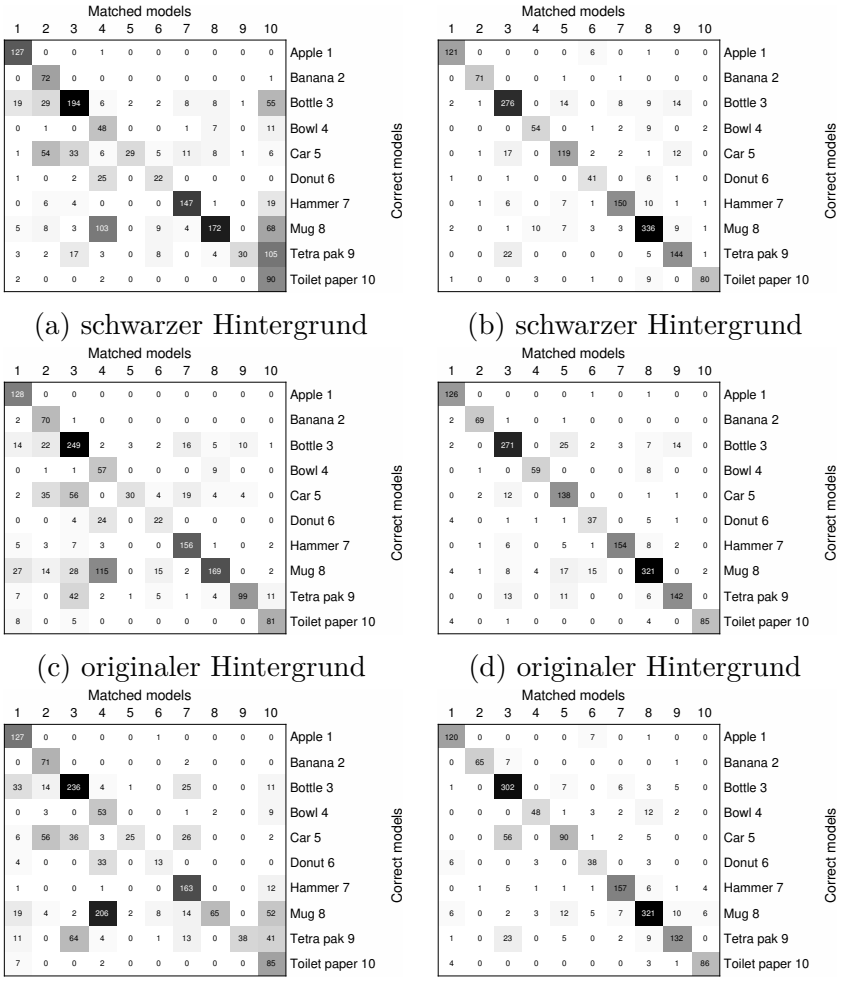
Tabelle 3.14: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.5.1 zeigt die CMs bei iterativem Training bei einer Konfidenz unter 80%. Hier ist bei schwarzem und weißem Hintergrund durch *self-check* eine Verbesserung in den CMs(abgeschwächter Bias) und in den Accuracies(ca. 2%) zu erkennen. Beim originalen Hintergrund wird zwar der grundlegende Bias ausgebessert, dafür werden, wie auch bei den beiden anderen Hintergründen, Objekte aufgrund der Augmentierung verlernt, was die Ergebnisse besonders hier schmälert. Die Accuracy konnte beim originalen Hintergrund um ca. 1% gesteigert werden.

Bei iterativem Training, bei einer Konfidenz unter 90% zeigen sich in den CMs (siehe Abbildung 3.3.5.2) sowie den Accuracies teilweise unterschiedliche Ergebnisse für die drei Klassifizierer. Bei keinem der hier auftretenden Ergebnisse konnte der grundlegende Bias komplett ausgebessert werden, wie dies schon zuvor der Fall war. Zusätzlich ist hier beim 'weißen' Klassifizierer anders als dies beim Training ohne *self-check* der Fall war, eine Verschlechterung in der Accuracy um 3% sichtbar. Dies tritt aufgrund der Augmentierung der Daten vor dem Trainingsprozess auf. Scheinbar verändert die Augmentierung die Daten teilweise so deutlich, sodass sie den Daten der Testdatenbank zu unähnlich werden.

Auch hier wurde wie bei anderen durchgeführten Lernprozessen anschließend an den Trainingsprozess die Test- und Trainingsdatenbank erneut klassifiziert

um hier Daten zu erhalten, während sich der jeweilige Klassifizierer nicht verändert.

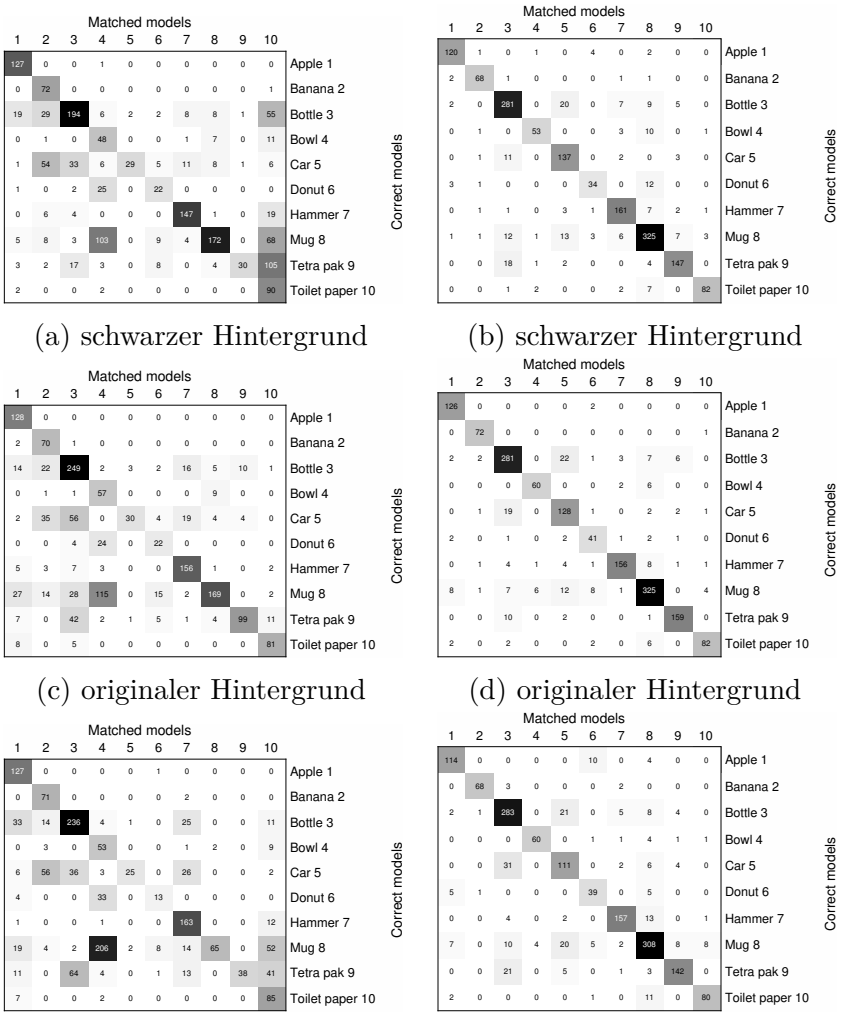


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.5.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.



(a) schwarzer Hintergrund

(b) schwarzer Hintergrund

(c) originaler Hintergrund

(d) originaler Hintergrund

(e) weißer Hintergrund

(f) weißer Hintergrund

Abbildung 3.3.5.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.

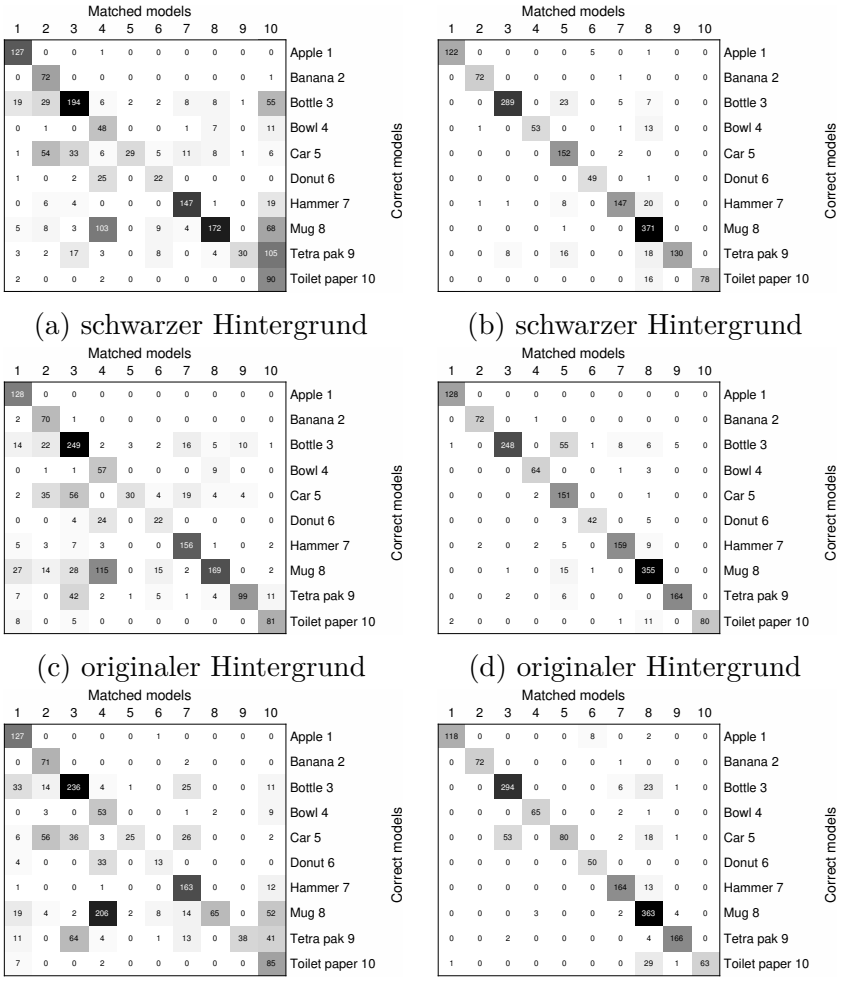
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	88,83%	94,60%
original	65,81%	90,75%	94,35%
weiß	54,34%	89,01%	89,70%

Tabelle 3.15: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.5.3 zeigt die CMs der Evaluierung der Testdatenbank bei iterativem Training bei einer Konfidenz unter 80%. Hier nimmt die Accuracy bei allen drei Klassifizierern gegenüber dem Training ohne *self-check* leicht ab(ca. 2%). Allerdings konnten die Bias-Fehler größtenteils ausgebessert werden. Lediglich Objekt 5 weißt beim 'schwarzen' und 'originalen' einen Bias auf und Objekt 8 beim 'weißen' Klassifizierer. Gründe hierfür liegen auch hier in der Augmentierung.

Bei iterativem Training, bei einer Konfidenz unter 90% zeigen die CMs(siehe Abbildung 3.3.5.3) teilweise bessere Ergebnisse als bei iterativem Training, bei einer Konfidenz unter 80%. Beim 'schwarzen' und 'originalen' Klassifizierer ist kein deutlicher Bias mehr zu erkennen und die Accuracy konnte um ca. 6 bzw. 4% gesteigert werden. Beim 'weißen' weist Objekt 5 einen schwächeren Bias als bei Training ohne *self-check* auf und die Accuracy konnte um ca. 3% gesteigert werden. Ebenfalls anders als beim Training ohne *self-check* konnte hier die Accuracy des 'weißen' Klassifizierers gehalten werden.

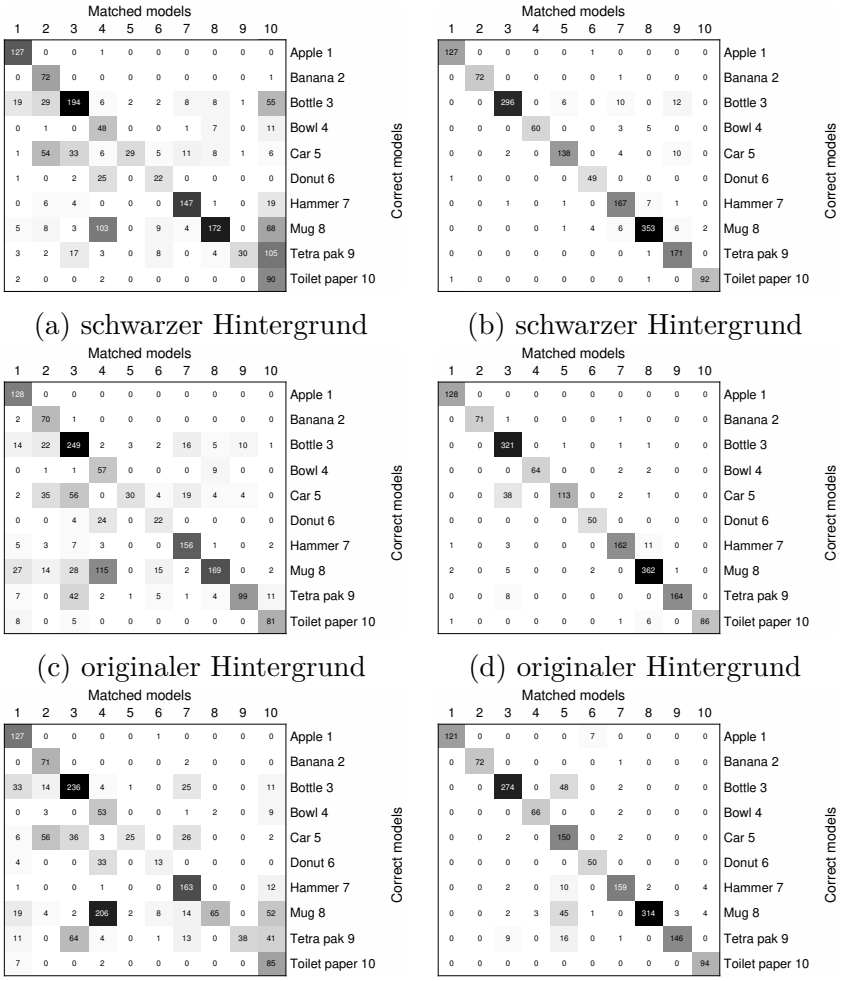


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.5.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.5.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.

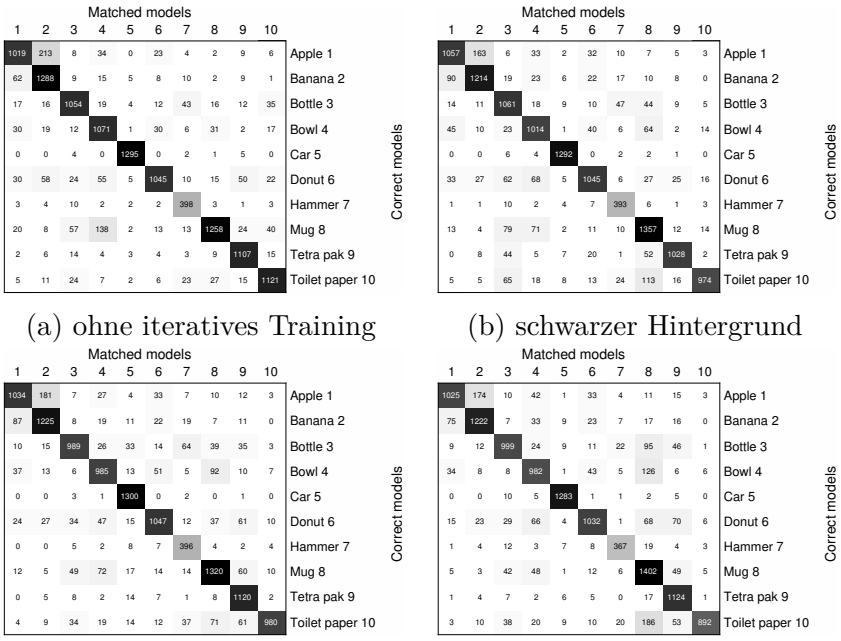
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz $< 80\%$	Konfidenz $< 90\%$
schwarz	87,31%	85,50%	85,21%
original	87,31%	85,18%	85,16%
weiß	87,31%	84,62%	86,02%

Tabelle 3.16: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.5.5 zeigt die CMs der Trainingsdatenbank bei iterativem Training bei einer Konfidenz unter 80%. Hier konnten durch *self-check* Bias-Fehler besser vermieden werden. Allerdings konnten diese nicht vollkommen verhindert werden. Die Accuracy hat sich im Vergleich zu Training ohne *self-check* um ca. $1 - 5\%$ verbessert werden. Allerdings ist auch hier die Accuracy gegenüber einer Evaluierung ohne iterativem Training um ca. $2 - 3\%$ gesunken.

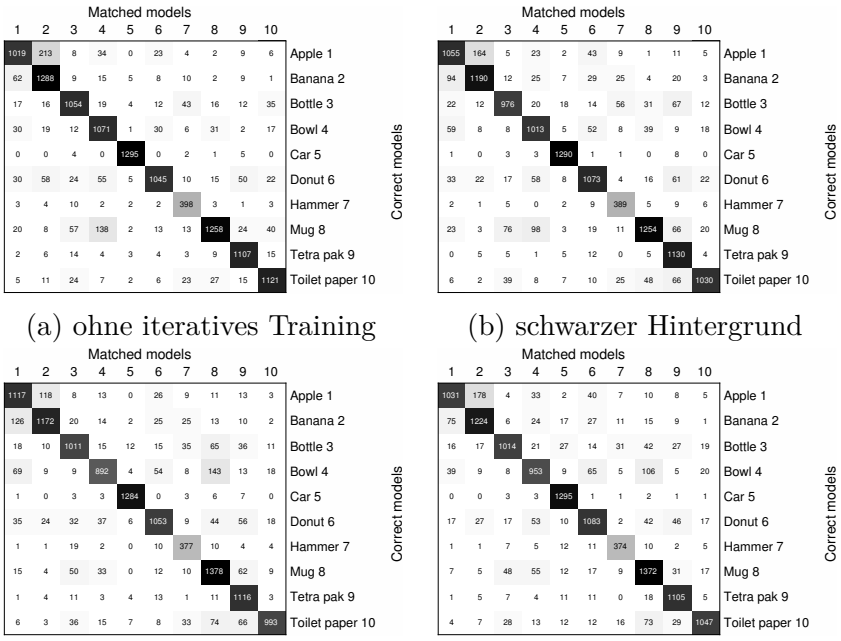
Bei iterativem Training, bei einer Konfidenz unter 90% zeigen die CMs(siehe Abbildung 3.3.5.6) erneut ähnliche Ergebnisse wie bei iterativem Trainings bei einer Konfidenz unter 80%. Die Accuracy konnte gehalten bzw. beim 'weißen' sogar um ca. 2% verbessert werden.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.5.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.5.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei iterativen Training und mit *self-check*.

3.3.6 Komplette Augmentierung und mit *self-check*

Bei den hier präsentierten Ergebnissen wurden die Daten bei der Klassifizierung zufällig augmentiert und auf diesen Daten trainiert. Dies ermöglicht das erneute Evaluieren des Algorithmus auf den Daten. Auch hier wurde *self-check* angewandt. In den Tabellen 3.17, 3.18 und 3.19 sind die Accuracies der jeweiligen Klassifizierungen vor und nach dem iterativen Training aufgelistet.

Klassifizierungen im Rahmen der Trainingsprozesse

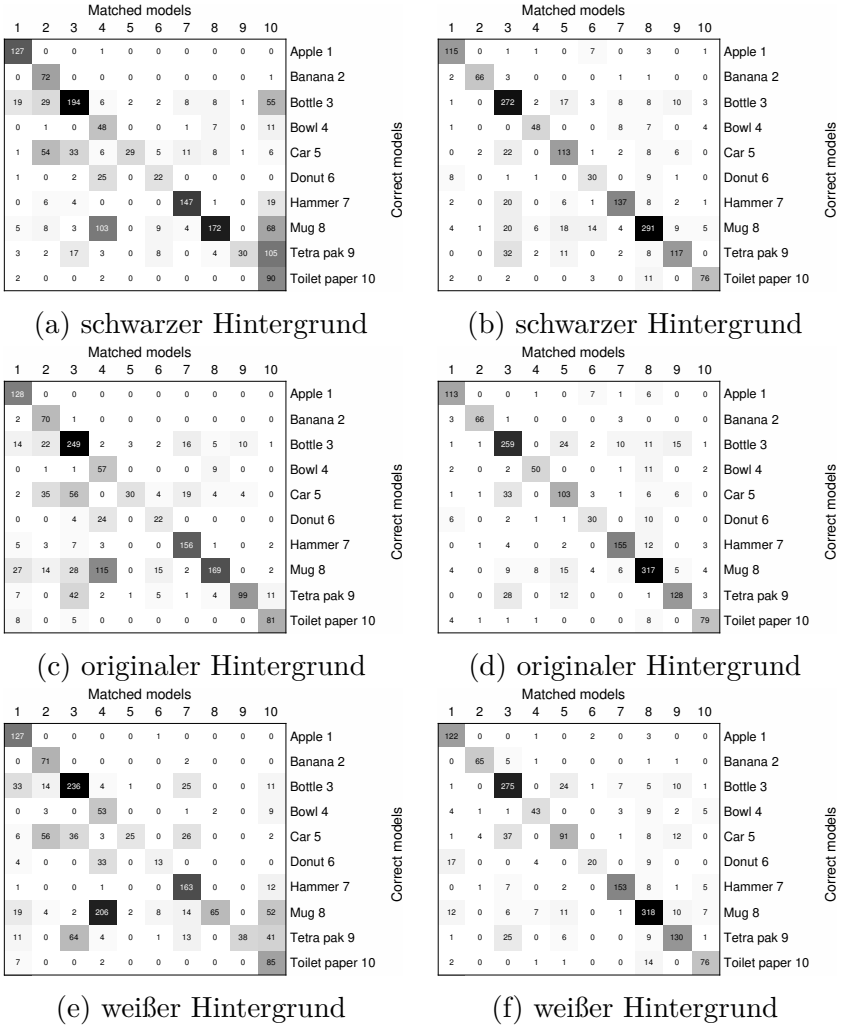
Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	78,47%	79,65%
original	65,81%	80,64%	84,49%
weiß	54,34%	80,21%	80,83%

Tabelle 3.17: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.6.1 zeigt die CMs bei iterativem Training bei einer Konfidenz unter 80%. Hier zeigen die CMs schwächere Bias-Fehler als bei Training ohne *self-check*. Die Accuracy hat sich beim schwarzen Hintergrund um ca. 0,4% und beim weißen um ca. 1% verbessert und beim originalen um ca. 1% verschlechtert. Der Grund warum die Bias-Fehler hier nicht mehr verhindert werden können, ist die Veränderung der Daten durch die Augmentierung.

Bei iterativem Training, bei einer Konfidenz unter 90% zeigen Die CMs(siehe Abbildung 3.3.6.2) ein ähnliches Bild wie bei Training unter einer Konfidenz von 80%. Die Accuracy zeigt allerdings eine Verbesserung von ca. 4% beim originalem Hintergrund, während die anderen beiden nur eine Verbesserung von unter 1% aufweisen. In diesem Fall konnten Bias-Fehler Aufgrund der Augmentierung bei der Klassifizierung nicht mehr so gut vermieden werden.

Auch hier wurde wie bei anderen durchgeführten Lernprozessen anschließend an den Trainingsprozess die Test- und Trainingsdatenbank erneut klassifiziert um hier Daten zu erhalten, während sich der jeweilige Klassifizierer nicht verändert und die Daten bei der Klassifizierung nicht augmentiert sind.

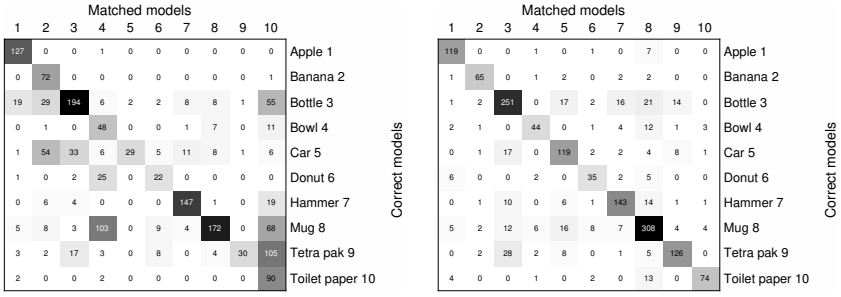


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

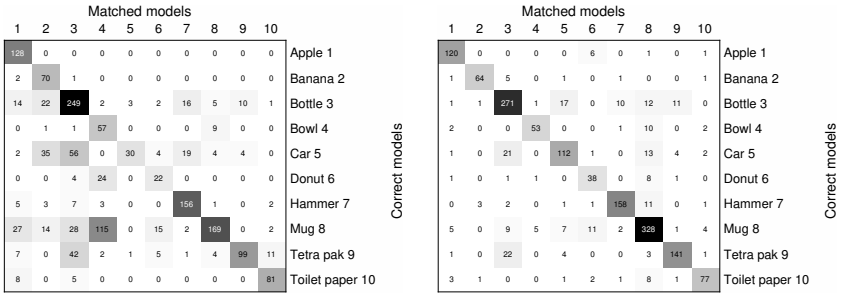
(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.6.1: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.



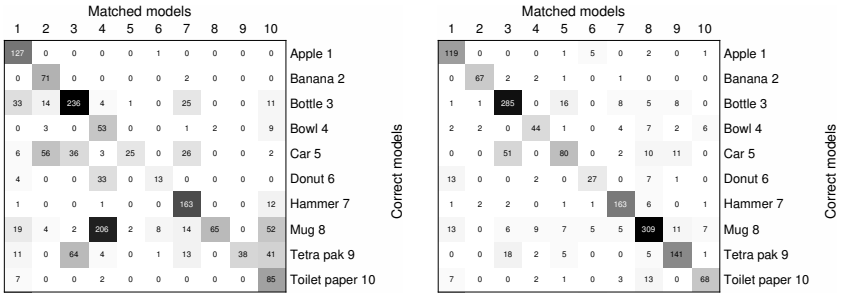
(a) schwarzer Hintergrund

(b) schwarzer Hintergrund



(c) originaler Hintergrund

(d) originaler Hintergrund



(e) weißer Hintergrund

(f) weißer Hintergrund

Abbildung 3.3.6.2: Confusion Matrices vor(links) bzw. nach(rechts) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.

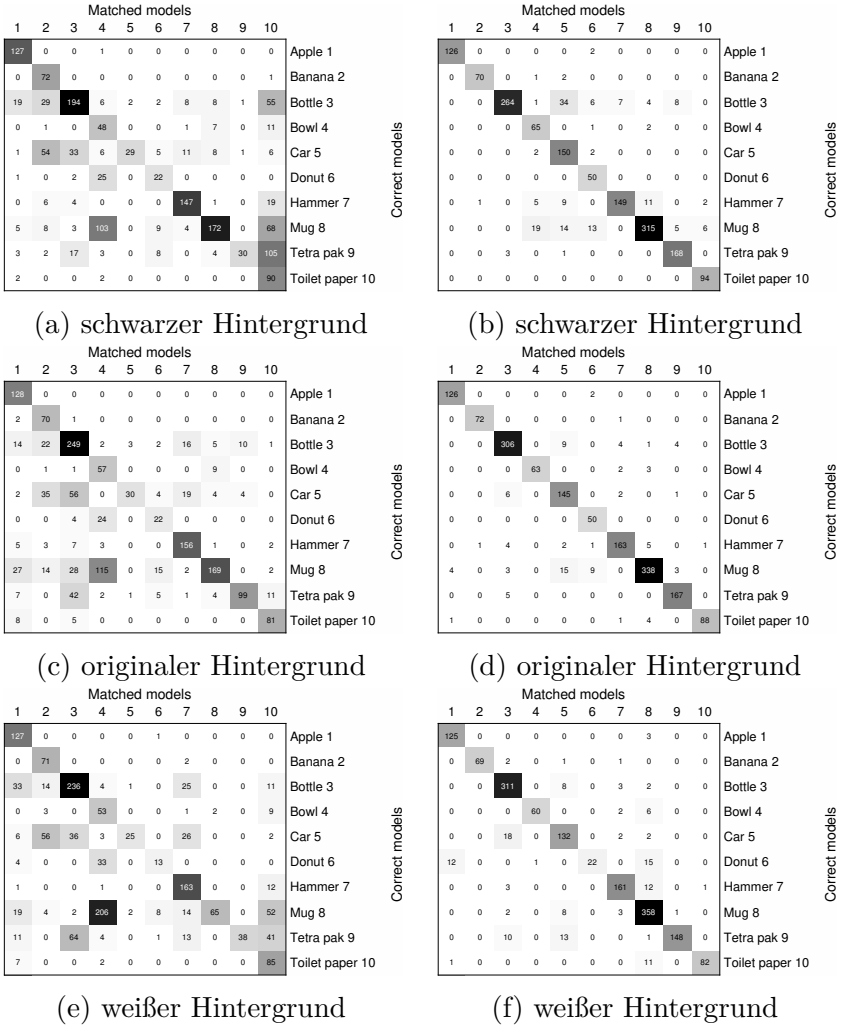
Evaluierung auf der Testdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	57,75%	90,01%	93,54%
original	65,81%	94,16%	93,79%
weiß	54,34%	91,06%	93,67%

Tabelle 3.18: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.6.3 zeigt die CMs der Evaluierung der Testdatenbank bei iterativem Training bei einer Konfidenz unter 80%. Die CMs zeigen hier trotz der teilweise schlechteren Ergebnisse bei den Trainingsprozessen immer noch gute Ergebnisse. Der Bias bei den Klassifizierern ist gering, lediglich vereinzelte Objekte sind nicht gut gelernt, z. B. Objekte 3, 7 und 8 beim 'schwarzen', Objekt 8 beim 'originalen' und Objekt 6 und 9 beim 'weißen' Klassifizierer. Dies ist auf die Augmentierung bei der Klassifizierung beim Trainingsprozess zurückzuführen.

Bei iterativem Training, bei einer Konfidenz unter 90% zeigt sich in den CMs(siehe Abbildung 3.3.6.4) erneut ein geringer Bias. Die Objekte werden hier im Gegensatz zum Training bei einer Konfidenz unter 80% besser gelernt und somit nicht so leicht verwechselt.

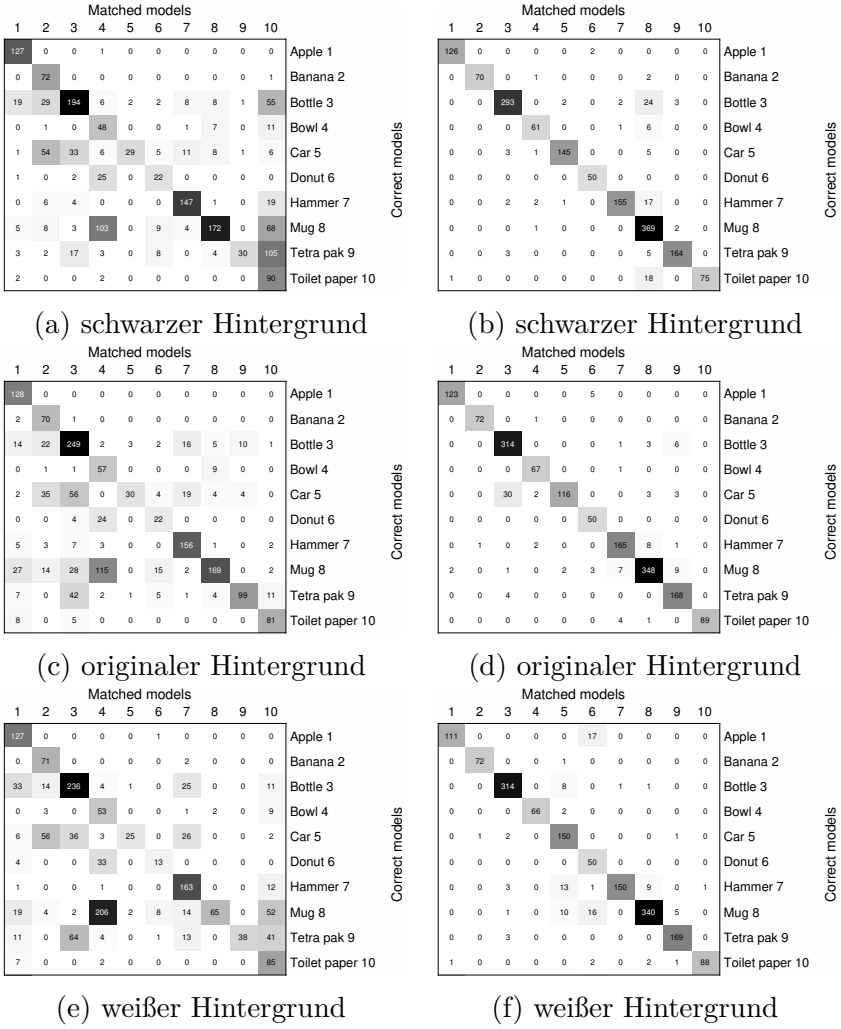


(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.6.3: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.



(a) schwarzer Hintergrund (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) originaler Hintergrund

(e) weißer Hintergrund (f) weißer Hintergrund

Abbildung 3.3.6.4: Confusion Matrices der Testdatenbank vor(links) bzw. nach(rechts) den Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.

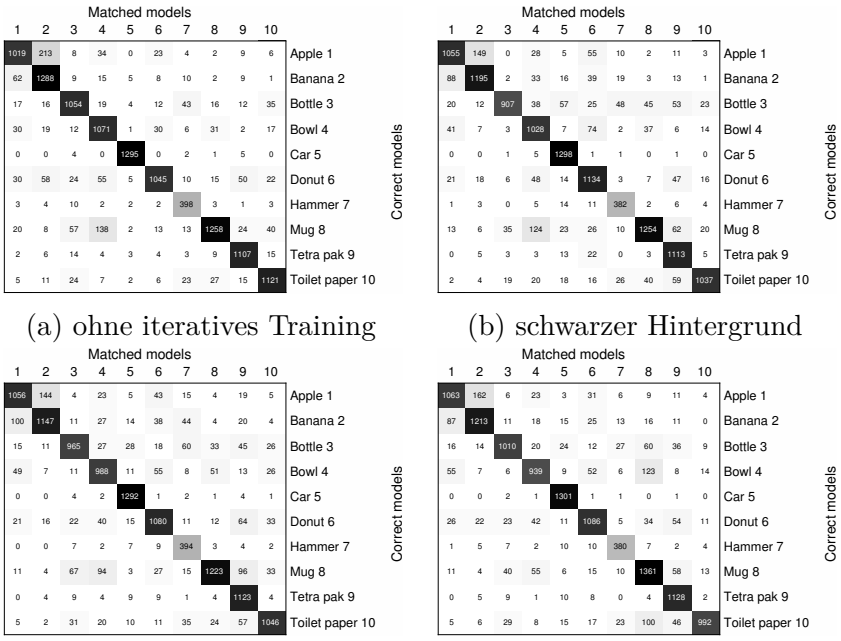
Evaluierung auf der Trainingsdatenbank

Hintergrundfarbe	ohne Training	Konfidenz < 80%	Konfidenz < 90%
schwarz	87,31%	85,24%	83,73%
original	87,31%	84,51%	84,05%
weiß	87,31%	85,81%	85,27%

Tabelle 3.19: Accuracy der unterschiedlichen Hintergründe mit unterschiedlichen Konfidenzen

Abbildung 3.3.6.5 zeigt die CMs der Trainingsdatenbank bei iterativem Training bei einer Konfidenz unter 80%. Hier weisen die drei Klassifizierer in ihren CMs ebenfalls einen eher geringen Bias auf. Allerdings sind Objekte im Vergleich zu keinem iterativen Training verlernt worden, z. B. Objekt 3 beim 'schwarzen', Objekte 3 und 4 beim 'originalen' und Objekt 4 beim 'weißen'. Der Grund hierfür ist ebenfalls die Augmentierung.

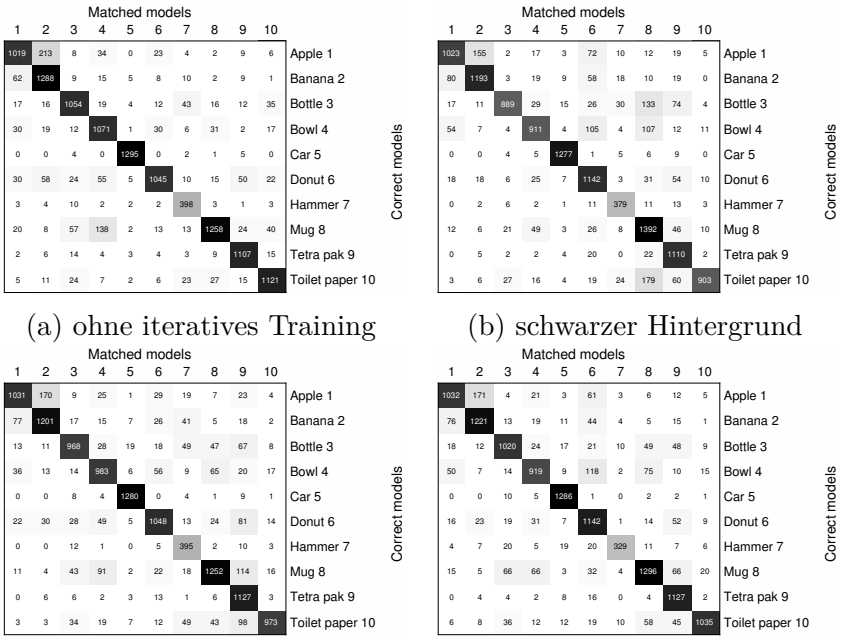
Bei iterativem Training, bei einer Konfidenz unter 90% zeigen die CMs(siehe Abbildung 3.3.6.6) anders als bei der Testdatenbank zeigen hier nicht den Effekt, dass die Objekte besser gelernt werden. Das zusätzliche Training bringt demnach durch die Augmentierung keine bessere Klassifizierung dieser Datenbank mit sich.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.6.5: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 80% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.



(a) ohne iteratives Training (b) schwarzer Hintergrund

(c) originaler Hintergrund (d) weißer Hintergrund

Abbildung 3.3.6.6: Confusion Matrices der Trainingsdatenbank vor (a) bzw. nach (b,c,d) den unterschiedlichen Trainingsprozessen unter 90% Konfidenz mit Augmentierung bei Klassifizierung und mit *self-check*.

4 Zusammenfassung

Diese Arbeit beschäftigte sich mit dem iterativen Verbessern einer Klassifizierung von Objekten für Roboter. Dies ist ein sehr nützlicher Vorgang, da so der Roboter zu Beginn seiner Aufgabe noch nicht alle Klassen bzw. Objekte kennen muss, sondern diese erst nach und nach lernen kann. Das iterative Verbessern wurde mit einem State-of-the-art Klassifizierungsalgoritmus, einem neuronalen Netz, durchgeführt. Das hier verwendete Netz basiert auf AlexNet[3], wurde allerdings an die hier verwendeten Datenbanken angepasst. Als Trainingsdatenbank diente eine Auswahl der ImageNet[4] Datenbank. Als Testdatenbank, wurden Bilder, mit unterschiedlichen Hintergründen, welche aus den Pointclouds der Cat-10 Testdatenbank des 3DNet[31] Datensatzes erstellt wurden, verwendet.

Zunächst wurde untersucht ob die Verbesserung des neuronalen Netzes überhaupt möglich ist, bzw. mit welchen Parametern dies am ehesten gelingt.

Frühe Untersuchungen zeigten, dass diese iterativen Trainingsprozesse leicht Bias-Fehler hervorrufen können. Daher wurde neben dem iterativen Trainingsprozess *self-check* zur Evaluierung des Bias und gegebenenfalls dessen Korrektur entwickelt und ebenfalls evaluiert.

Die entwickelte Methode *self-check* zur der Bias-Korrektur erforderte Augmentierung der Bilddaten. Es wurden 11 verschiedene Augmentierungsarten gewählt und neben *self-check* auch beim iterativen Lernen in unterschiedlichen Momenten des Klassifizierungs- bzw. Trainingsprozesses zur Untersuchung angewandt.

Die Ergebnisse der unterschiedlichen iterativen Trainingsprozesse zeigten, dass eine deutliche Verbesserung(min. $20 - 30\%$) durch iterative Trainings des Klassifizierers und so ein Anpassen desselben an die zu evaluierenden Daten möglich ist. Die unterschiedlichen Hintergründe zeigen alle eine Verbesserung, wobei die Klassifizierer, welche auf den Bildern mit originalem Hintergrund trainiert wurden im Durchschnitt die höchste Anzahl an korrekten Klassifizierungen liefern. Hier zeigen die Klassifizierer, dass sie sich sehr gut an die Bilder anpassen, allerdings die schon trainierten Daten verlernen. Dies konnte im Rahmen dieser Arbeit durch *self-check* weitgehend korrigiert werden.

Die Klassifizierer, welche auf Bildern mit schwarzem Hintergrund trainiert wurden, zeigten ebenfalls sehr gute Ergebnisse. Bei diesen Klassifizieren ist die Neigung Daten zu verlernen geringer, als bei originalem Hintergrund. Bei

diesen, sowie den Klassifizierern, welche mit weißem Hintergrund arbeiten hängt die Performance auch stark mit der Segmentierung, welche im Zuge der Bilderstellung verwendet wurde zusammen. Diese wurde allerdings im Rahmen dieser Arbeit nicht untersucht und könnte für weiter Untersuchungen interessant sein.

Die Klassifizierer, welche mit weißem Hintergrund arbeiten neigten am ehesten zu Bias-Fehlern. Dies kann auf die Objekte der Testdatenbank zurückzuführen sein, da in dieser zahlreiche weiße Objekte vorkommen. Hier könnten genauere Untersuchungen der Auswirkungen des Hintergrundes mit anderen Objekten Aufschluss über Auswirkung der Farbe geben.

Die Augmentierung hatte während des iterativen Trainingsprozesses nur einen geringen Einfluss auf die CMs bzw. die Accuracy. Die Augmentierung im Rahmen des Klassifizierungsprozesses führte vor allem bei den CMs der Trainingsdatenbank zu schlechteren Ergebnissen, da sich durch die Augmentierung die Bilder scheinbar stärker verändern. Die Augmentierung zeigte ihren Einfluss unabhängig von der Art des Hintergrundes.

Die von uns entwickelte *self-check* Methode zur Bias-Korrektur verbessert die Ergebnisse der Klassifizierer bei Evaluierung auf der Test- und der Trainingsdatenbank.

4.1 Ausblick

Im Rahmen dieser Arbeit konnten viele sehr unterschiedliche Experimente durchgeführt werden. Hier könnten weiter Arbeiten die Ergebnisse verfeinern:

- Weitere Untersuchung der Auswirkung des Hintergrunds bezüglich unterschiedlicher Objekte und iterativem Training, um zu untersuchen ob die schlechtere Performance des weißen Klassifizierers auf die Objekte in der Testdatenbank zurückzuführen ist.
- Evaluierung auf zusätzlichen größeren variantenreicheren Testsets, um die Verallgemeinerungsfähigkeit der hier entwickelten Klassifizierer weiter zu testen.
- Veränderung der Segmentierungsparameter, um festzustellen ob so eine bessere Accuracy durch den schwarzen bzw. weißen Klassifizierer erbracht werden kann.

Weiters können weiterführende Arbeiten die hier gebrachten Entwicklungen weiter vorantreiben:

- Implementierung des Klassifizierers in einem System mit Kamera und längere Tests mit Benutzern, um Langzeitauswirkungen des iterativen Trainings zu ermitteln. Hierzu könnten die in [27] entwickelte Methode hilfreich sein, um die Dauer des Trainingsprozesses zu reduzieren.
- AlexNet könnte lediglich zur Errechnung von Deskriptoren verwendet werden. Als Klassifizierer könnte einer der in 1.2.2 vorgestellten Algorithmen herangezogen werden. Dies könnte die Trainingszeit reduzieren sowie, die Integration von anderen Deskriptoren in den Klassifizierungsprozess ermöglichen.
- Es könnte der letzte und vorletzte Layer von AlexNet neu trainiert werden um dem Klassifizierer die Möglichkeit zu geben komplexere Schlüsse zu ziehen.

Literatur

[1] P. Baldi und K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima", *Neural networks*, Bd. 2, Nr. 1, S. 53–58, 1989.

[2] "Neuronale netze: Grundlagen", in *Soft Computing in der Bioinformatik: Eine grundlegende Einführung und Übersicht*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 27–41, ISBN: 978-3-540-29887-8. Adresse: `http://dx.doi.org/10.1007/3-540-29887-8_3`.

[3] A. Krizhevsky, I. Sutskever und G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger, Hrsg., Curran Associates, Inc., 2012, S. 1097–1105. Adresse: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg und L. Fei-Fei, "Imagenet large scale visual recognition challenge", *International Journal of Computer Vision (IJCV)*, Bd. 115, Nr. 3, S. 211–252, 2015.

[5] M. Muja und D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration", in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, INSTICC Press, 2009, S. 331–340.

[6] C.-C. Chang und C.-J. Lin, "Libsvm: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, Bd. 2, 27:1–27:27, 3 2011.

[7] J. Sun, G. Zhong, J. Dong, H. Saeeda und Q. Zhang, "Cooperative profit random forests with application in ocean front recognition", *IEEE Access*, Bd. 5, S. 1398–1408, 2017, ISSN: 2169-3536.

[8] R. A. Nugrahaeni und K. Mutijarsa, "Comparative analysis of machine learning knn, svm, and random forests algorithm for facial expression classification", in *2016 International Seminar on Application for Technology of Information and Communication (ISemantic)*, 2016, S. 163–168.

[9] T. Fäulhammer, M. Zillich, J. Prankl und M. Vincze, "A multi-modal rgb-d object recognizer", in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, S. 733–738.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, Bd. 60, Nr. 2, S. 91–110, 2004, ISSN: 1573-1405. Adresse: `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`.

[11] F. Tombari, S. Salti und L. D. Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching", in *2011 18th IEEE International Conference on Image Processing*, 2011, S. 809–812.

[12] W. Wohlkinger und M. Vincze, "Ensemble of shape functions for 3d object classification", in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, 2011, S. 2987–2992.

[13] D. N. Vasundhara und M. Seetha, "Rough-set and artificial neural networks based image classification", in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016, S. 35–39.

[14] M. A. Ibrahim, M. K. Arora, S. K. Ghosh und H. Chen, "Approaches to improve accuracy of neural network classification of images dominated by mixed pixels", in *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, Bd. 1, 2004, S. 571.

[15] A. Kumar, J. Kim, D. Lyndon, M. Fulham und D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification", *IEEE Journal of Biomedical and Health Informatics*, Bd. 21, Nr. 1, S. 31–40, 2017, ISSN: 2168-2194.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke und A. Rabinovich, "Going deeper with convolutions", in *Computer Vision and Pattern Recognition (CVPR)*, 2015. Adresse: `http://arxiv.org/abs/1409.4842`.

[17] Y. Liu, C. J. Sullivan und F. d'Errico, "Machine learning method applied in readout system of superheated droplet detector", *IEEE Transactions on Nuclear Science*, Bd. PP, Nr. 99, S. 1–1, 2017, ISSN: 0018-9499.

[18] N. A. Syed, S. Huan, L. Kah und K. Sung, "Incremental learning with support vector machines", 1999.

[19] Z. Akata, F. Perronnin, Z. Harchaoui und C. Schmid, "Good practice in large-scale learning for image classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 36, Nr. 3, S. 507–520, 2014, ISSN: 0162-8828.

[20] H. Chen und W. Wei, "Pseudo-example based iterative svm learning approach for gender classification", in *2006 6th World Congress on Intelligent Control and Automation*, Bd. 2, 2006, S. 9528–9532.

[21] J. Lei, G. Li, J. Zhang, D. Lu und Q. Guo, "Online structural svm learning by dual ascending procedure", in *Proceedings 2014 IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 2014, S. 212–217.

[22] D. M. J. Tax und P. Laskov, "Online svm learning: From classification to data description and back", in *2003 IEEE XIII Workshop on Neural Networks for Signal Processing (IEEE Cat. No.03TH8718)*, 2003, S. 499–508.

[23] R. Polikar, L. Upda, S. S. Upda und V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks", *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, Bd. 31, Nr. 4, S. 497–508, 2001.

[24] S. S. Bharadwaj, R. C. Kumar, B. N. Sumukha und K. George, "A self-monitoring online sequential learning mechanism for feedfoward neural networks", in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016, S. 23–28.

[25] Y. Cui, C. Surpur, S. Ahmad und J. Hawkins, "A comparative study of htm and other neural network models for online sequence learning with streaming data", in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, S. 1530–1538.

[26] A. D. M. Garvin und P. J. W. Rayner, "Fist: Fast iterative self-structuring and training of artificial neural networks", in *Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN '94., 1994 International Symposium on*, 1994, 377–380 vol.2.

[27] K. Wang, P. Guo, Q. Yin, A. L. Luo und X. Xin, "A pseudoinverse incremental algorithm for fast training deep neural networks with application to spectra pattern recognition", in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, S. 3453–3460.

[28] D. Liu, T.-S. Chang und Y. Zhang, "A constructive algorithm for feedforward neural networks with incremental training", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Bd. 49, Nr. 12, S. 1876–1879, 2002, ISSN: 1057-7122.

[29] G. Cheng, T. Liu, X. Wang und Q. Huang, "Rapid training for self-organizing neural networks with incremental", in *Sixth International Conference on Intelligent Systems Design and Applications*, Bd. 1, 2006, S. 28–31.

[30] I. P. Morns und S. S. Dlay, "The dynamic supervised forward-propagation neural network for handwritten character recognition using fourier descriptors and incremental training", in *Proceedings of Third International Conference on Electronics, Circuits, and Systems*, Bd. 2, 1996, 1123–1126 vol.2.

[31] W. Wohlkinger, A. Aldoma Buchaca, R. Rusu und M. Vincze, "3dnet: large-scale object class recognition from cad models", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[32] M. D. Zeiler und R. Fergus, "Visualizing and understanding convolutional networks", in *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, D. Fleet, T. Pajdla, B. Schiele und T. Tuytelaars, Hrsg. Cham: Springer International Publishing, 2014, S. 818–833, ISBN: 978-3-319-10590-1. Adresse: http://dx.doi.org/10.1007/978-3-319-10590-1_53.

[33] S. Gupta, R. Girshick, P. Arbeláez und J. Malik, "Learning rich features from rgb-d images for object detection and segmentation", in *European Conference on Computer Vision*, Springer, 2014, S. 345–360.

[34] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama und T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", *ArXiv preprint arXiv:1408.5093*, 2014.

[35] R. Hecht-Nielsen, "Theory of the backpropagation neural network", in *International 1989 Joint Conference on Neural Networks*, 1989, 593–605 vol.1.

[36] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms", in *Proceedings of the Twenty-first International Conference on Machine Learning*, Ser. ICML '04, Banff, Alberta, Canada: ACM, 2004, S. 116–, ISBN: 1-58113-838-5. Adresse: http://doi.acm.org/10.1145/1015330.1015332.

A Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, im Mai 2017

Martin Palkovits