|  |  |  |
| --- | --- | --- |
| **MA 3062** | | |
| Untersuchung und prototypische Umsetzung eines Lifelong Deep Neural Network Algorithmus | | |
| **Simon Kamm** | | |
|  | | |
| **Lastenheft** | | |
|  | Prüfer: | Prof. Dr.-Ing. Michael Weyrich |
|  | Betreuer: | Benjamin Maschler, M.Sc. |
| Start: 29.04.2019 | | Abgabe: 29.10.2019 |
|  | |  |

**Dokument Versionsverwaltung**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Autor | QS | Datum | Status | Änderungen |
| 0.1 | Kamm | Ms | 22.08.19 | in Bearb. | Erstellung |
| 1.0 | Kamm | Ms | 28.08.19 | Vorgelegt | Erste Version Evaluierungsprotokoll (ohne ImageNet) |
| 1.1 | Kamm | Ms | 07.10.19 | Vorgelegt | Zusätzlicher Testfall Consolidation und ImageNet-Full |

# Inhaltsverzeichnis

0 Inhaltsverzeichnis 2

1 Generelle Anmerkungen Evaluierungsprotokoll 4

2 Hyperparameter-Optimierung Modul B 6

2.1 Bezug zur Evaluierungsspezifikation 6

2.2 Evaluierungsprozedur 6

2.3 Ergebnis 7

2.4 Auswertung 9

3 Anzahl an Trainingsbilder pro Klasse 12

3.1 Bezug zur Evaluierungsspezifikation 12

3.2 Evaluierungsprozedur 12

3.3 Ergebnis 12

3.4 Auswertung 14

4 Untersuchung Continual Learning auf Testdaten 16

4.1 Bezug zur Evaluierungsspezifikation 16

4.2 Evaluierungsprozedur 16

4.3 Ergebnis 16

4.4 Auswertung 17

5 Untersuchung Distributed Learning auf Testdaten 19

5.1 Bezug zur Evaluierungsspezifikation 19

5.2 Evaluierungsprozedur 19

5.3 Ergebnis 19

5.4 Auswertung 20

6 Einfluss von Konsolidierungsschritten 21

6.1 Bezug zur Evaluierungsspezifikation 21

6.2 Evaluierungsprozedur 21

6.3 Ergebnis 21

6.4 Auswertung 23

7 ImageNet 25

7.1 Bezug zur Evaluierungsspezifikation 25

7.2 Evaluierungsprozedur 25

7.3 Ergebnis 25

7.4 Auswertung 27

8 Literaturverzeichnis 29

# Generelle Anmerkungen Evaluierungsprotokoll

In diesem Kapitel werden einige grundlegende Anmerkungen geführt. Diese werden zentral zu Beginn aufgeführt, um eine bessere Lesbarkeit im weiteren Verlauf zu gewährleisten und um mehrfach auftretende Änderungen nur einmal zu erwähnen.

Für ImageNet-10 wurden Bilder der Dimension 64x64x3 genutzt, da dieser Datensatz bereits lokal in dieser Konfiguration vorhanden war. Eine Nutzung von 224x224 Bildern hätte einen erheblichen Zusatzaufwand bedeutet. Der Test auf dem gesamten ImageNet-Datensatz wird wie in der Evaluierungsspezifikation beschrieben auf Bildern der Dimensionen 224x224x3 durchgeführt haben.

Aufgrund des vortrainierten MobileNet-v2 konnten nur spezielle Eingangsdimensionen für die Bilder genutzt werden. Die kleinste verfügbare Dimension ist dabei 96x96. Deshalb wurden kleinere Bilder (ImageNet-10 und MNIST) auf diese Dimensionen mithilfe der TensorFlow-Funktion *tf.image.resize* vergrößert. Zusätzlich müssen die Bilder im RGB-Format vorliegen. Für ImageNet-Bilder ist dies der Fall. MNIST-Bilder werden mithilfe der TensorFlow-Funktion *tf.image.grayscale\_to\_rgb* umgewandelt. In Abbildung 1 ist ein Bild des MNIST-Datensatzes vor der Bild-Augmentation im Format 28x28x1 zu sehen. In diesem Format liegen die Bilder des Datensatz vor. Abbildung 1 zeigt die Zahl „7“.

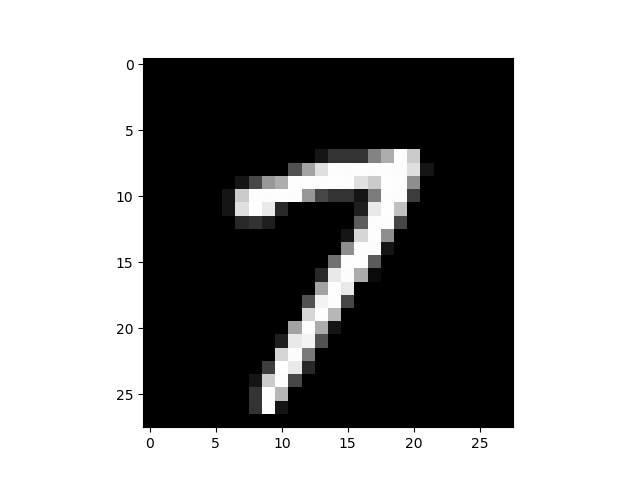


Abbildung 1: MNIST-Bild vor Bild-Augmentation

Abbildung 2 zeigt ein Bild des MNIST-Datensatzes nach der zuvor beschriebenen Bild-Augmentation. Es besitzt nun das Format 96x96x3. Jedoch ist zu sehen, dass das Bild in seiner Grundstruktur nicht wesentlich verändert wurde, wodurch durch diese Bild-Augmentation kein großer Einfluss auf Klassifikationsgenauigkeit erwartet wird. In diesem Beispielbild ist die Zahl „8“ zu sehen.

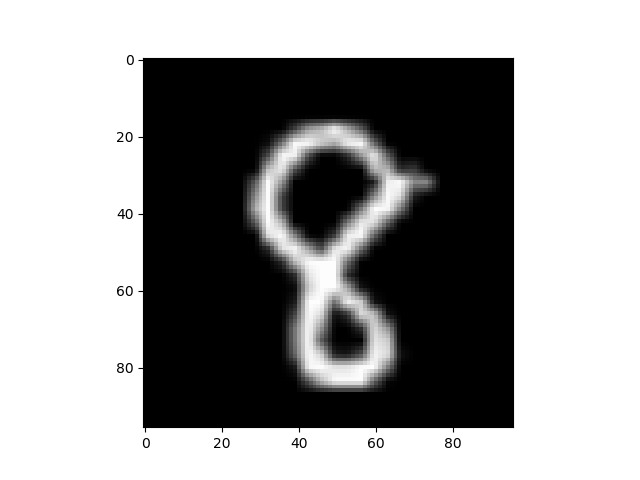


Abbildung 2: MNIST-Bild nach Augmentation

Zusätzlich wird die Bezeichnung des gesamten ImageNet-Datensatzes genauer definiert. Es wird der Datensatz der *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) aus dem Jahr 2012 genutzt [1]. Als Test-Daten werden wie in der Literatur und für vergleichbare Modelle üblich die frei verfügbaren 50 Validationsbilder pro Klasse genutzt. In dem genutzten Datensatz gibt es für die 1000 Klassen insgesamt 1.281.167 Trainingsbilder. Die Anzahl an Trainingsbilder pro Klasse schwankt dabei zwischen 732 und 1300. Mit 50 Validationsbildern (hier später als Test-Bilder genutzt und bezeichnet) pro Klasse sind 50.000 Validationsbilder vorhanden. Weitere Informationen über die unterschiedlichen Klassen können in [1] nachgelesen werden.

# Hyperparameter-Optimierung Modul B

Neuronale Netzwerke besitzen eine Vielzahl an Hyperparametern, welche abhängig vom konkreten Anwendungsfall und den vorliegenden Daten unterschiedlich eingestellt werden können und müssen für eine optimale Performanz des Netzwerkes. Um eine möglichst optimale Parametrierung des inkrementellen Klassifikators in Modul B zu erhalten, wurden die relevanten Parameter identifiziert und mithilfe einer Gitter-Suche (*Grid Search*) die optimalen Werte für die folgenden Evaluierungsfälle zu finden. Als relevante Parameter des FuzzyARTMAP-Netzwerkes wurden die Lernrate und der Vigilance-Parameter identifiziert, da diese das Training des Netzwerkes und damit am Ende die Performanz im Testfall beeinflussen. Details zu den beiden Modell-Parametern sind in der Konzeption zu finden.

## Bezug zur Evaluierungsspezifikation

Die in der Evaluierungsspezifikation beschriebenen Fälle des kontinuierlichen Lernens auf einem Endgerät für Split-MNIST und ImageNet-10 werden genutzt, um die oben beschriebenen Hyperparameter zu optimieren und deren Einfluss zu untersuchen. Diese beiden Datensätze erlauben es aufgrund ihres begrenzten Umfangs (lediglich 10 Klassen) viele Tests in kurzer Zeit durchzuführen, was für die Gitter-Suche relevant ist, da dort viele Tests für eine Abdeckung des Gitters erforderlich sind.

## Evaluierungsprozedur

Für die Hyperparameter-Optimierung wird eine zwei-dimensionale Gitter-Suche mit den Parametern und durchgeführt. Dafür werden diese beiden Parameter jeweils im Bereich in 0,1-er Schritten erhöht. Es werden 5 Wiederholungen pro mögliche Kombination durchgeführt, um eine statistische Aussagekraft zu erhalten. Das Vorgehen resultiert in jeweils 500 Tests für Split-MNIST und ImageNet-10. Die weiteren Parameter werden auf fixe Werte eingestellt, welche im Folgenden aufgelistet sind:

* modul\_b\_epsilon = 0,001
* modul\_b\_s = 1,05
* train\_img\_per\_class = 20
* test\_img\_per\_class = 100

Auf der Basis der erzielten Ergebnisse werden und ausgewählt.

Zu erwähnen ist, dass die Gruppen für Split-MNIST in einer festen Reihenfolge gezogen wurden (wie in der Evaluierungsspezifikation beschrieben). Bei ImageNet-10 wird die Reihenfolge der Klassen zufällig bestimmt, um eine zusätzliche Varianz der Trainingsdaten mit einzubeziehen.

Für beide Datensätze werden in jeder Wiederholung zufällig die benötigte Anzahl (20) an Trainingsbilder aus dem größeren Trainingsdatensatz gezogen. Die ermittelte Genauigkeit wird auf Validationsdaten aus dem Trainings-Datensatz bestimmt. Diese Bilder werden nicht für das Training verwendet. Durch die Wiederholung und das zufällige Ziehen der Trainings- und Validations-Samples findet eine Kreuz-Validation der Ergebnisse statt. Die Test-Datensätze werden dann im späteren Verlauf der Arbeit zur finalen Performanz-Bewertung eingesetzt.

Zur Umsetzung des Tests müssen lediglich die Parameter gesetzt werden und die korrekten Parameter in den jeweiligen *for-*Schleifen eingesetzt werden. Dann kann der beschriebene SW-Prototyp ausgeführt werden und die *Main*-Funktion iteriert automatisch über alle Testfälle und sichert die Ergebnisse in den passenden Ordnern ab.

Nachdem alle Tests durchlaufen sind, können mithilfe des zusätzlichen Auswerte-Skripts (*Evaluation\_Results.py*) die gewünschten Metriken und Graphen für die verschiedenen Fälle erstellt werden.

## Ergebnis

In diesem Kapitel werden die Ergebnisse für Split-MNIST und ImageNet-10 bewertet.

Es wird erwartet, dass für sehr kleine und große Werte von (Werte nahe 0 und 1) schlechte Ergebnisse erzielt werden, da bei diesen Fällen kaum ein Lernen stattfindet. Bei werden neue Trainingsdaten nicht genutzt und bei wird lediglich das neue Trainingssample genutzt und die alte Repräsentation damit überschrieben. Hier muss ein Wert gefunden werden, der es erlaubt altes, vorhandenes Wissen mit neuen Samples zu verbinden um eine möglichst ideale und generalisierte Repräsentation dieser Klasse zu bilden.

Bei wird erwartet, dass ab einem gewissen Wert keine Änderung mehr zu erkennen ist, da die Ähnlichkeit zwischen zwei Samples einer Klasse selten nahe 1 liegt. Dieser Schwellwert für wird für die unterschiedlichen Datensätze unterschiedlich erwartet. Ab diesem Schwellwert kann gesagt werden, dass jedes Sample, das im Training gesehen wird, als neue Repräsentation angelegt wird. Dies erfordert einen hohen Speicherbedarf. Zudem sind die erhaltenen Repräsentationen nicht generalisiert, wodurch auf späteren, abweichenden Test-Daten keine guten Generalisierungsfähigkeiten erwartet werden. Deshalb muss ein Wert für unterhalb dieser Schwelle gefunden werden. Ein zu kleiner Wert für führt dazu, dass die einzelnen Klassen durch sehr wenige (im Extremfall durch eine) Repräsentationen dargestellt werden. Je nach Klasse kann das positiv sein (wenn alle Samples der Klasse sehr ähnlich aussehen). Im Allgemeinen ist das aber nicht wünschenswert, da dadurch abweichende Samples der Klasse (zum Beispiel eine verdrehte Zahl) nicht gut erkannt werden können. Deshalb muss für ein guter Mittelwert zwischen zu vielen und zu wenigen Repräsentationen gewählt werden.

In Abbildung 3 sind die Ergebnisse der beschriebenen Gitter-Sucher zur Hyperparameter-Optimierung von Modul B auf Basis des Split-MNIST Datensatzes dargestellt. Entlang der x-Achse sind die unterschiedlichen Werte von zu sehen und entlang der y-Achse sind die Werte für aufgetragen. In den jeweiligen Feldern ist der Mittelwert der Klassifikationsgenauigkeit aus den fünf Wiederholungen für diese Parameterkombination eingetragen.

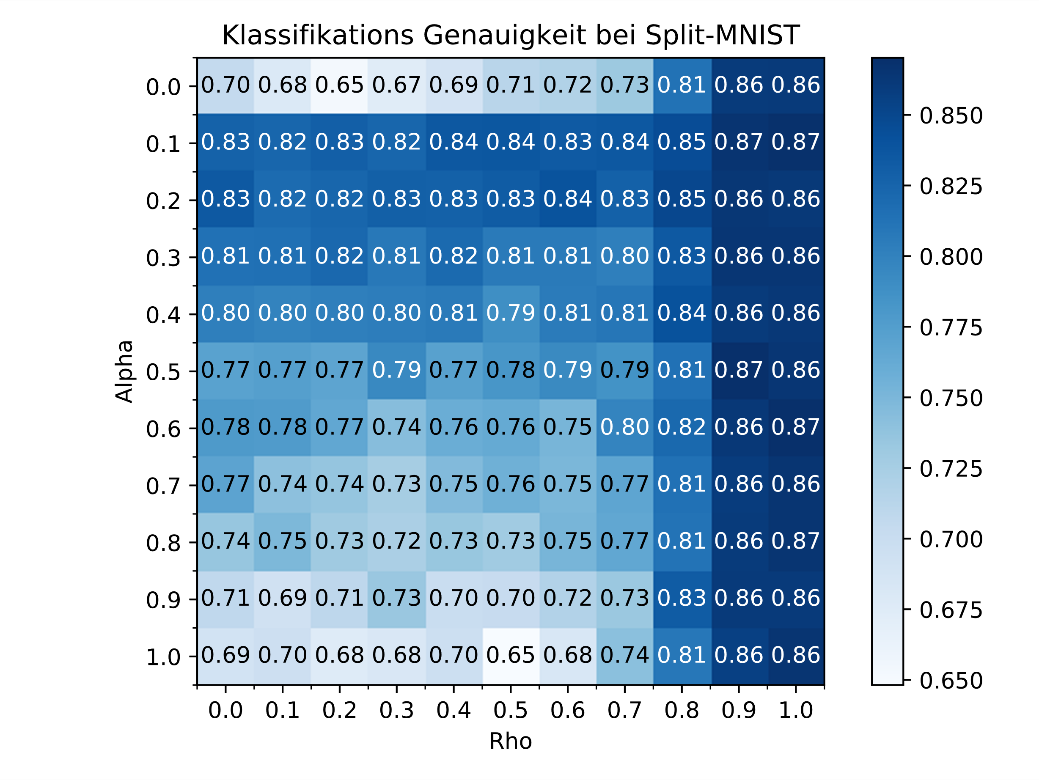


Abbildung 3: Ergebnisse der Gitter-Suche für und auf Basis des Split-MNIST Datensatzes

In Abbildung 4 sind die Ergebnisse für dieselben Testfälle auf Basis des ImageNet-10 Datensatzes dargestellt. Die Darstellungsart ist dabei identisch wie bereits für Abbildung 3 beschrieben.

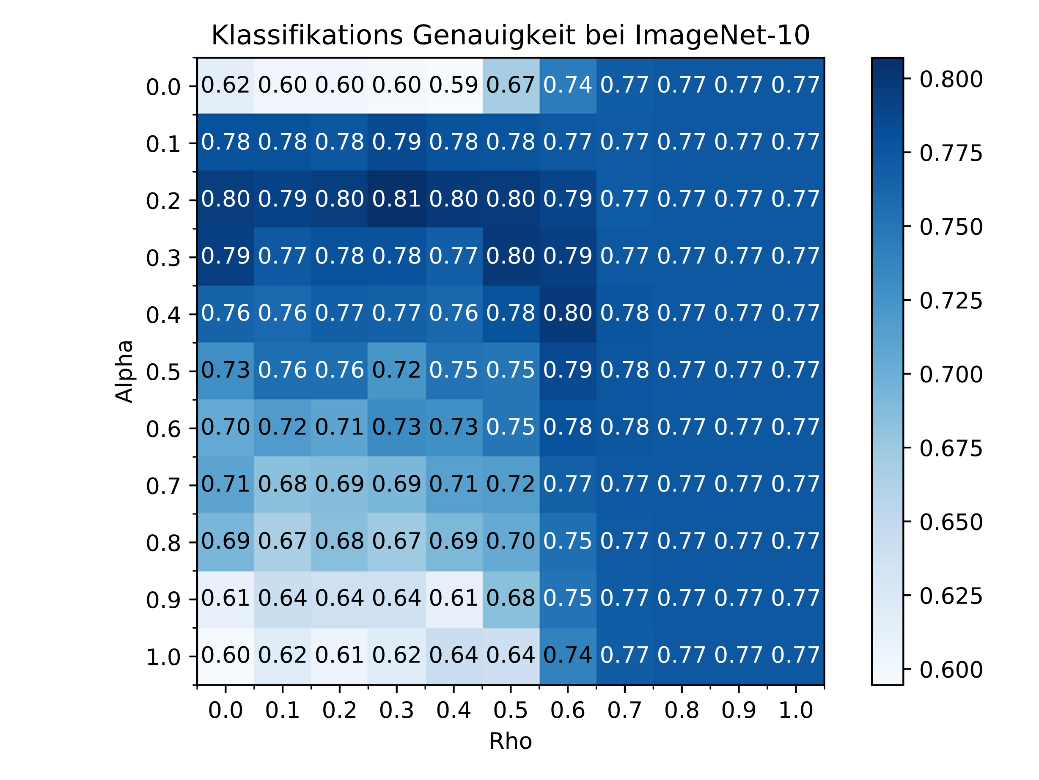


Abbildung 4: Ergebnisse der Gitter-Sucher für und auf Basis des ImageNet-10 Datensatzes

In Kapitel 2.4 werden die erzielten Ergebnisse nachfolgend ausgewertet und die passenden Hyperparameter für Modul B im Rahmen dieser Arbeit ausgewählt.

## Auswertung

Die Ergebnisse können für beide Parameter getrennt bewertet werden, da es keine relevante Korrelation der beiden Parameter gibt.

Zunächst werden die Ergebnisse für den Parameter bewertet. Hier ist das erwartete Verhalten zu erkennen. Bei beiden Datensätzen sind für hohe Werte von (>0,8) schlechte Ergebnisse zu sehen, da in diesem Fall lediglich die neuen Daten verwendet werden, um die Repräsentation zu bilden. Bei sind ebenfalls schlechte Ergebnisse zu beobachten, weil nur das erste gesehene Sample genutzt wird, um die Repräsentation zu bilden. Anhand der in Abbildung 3 und Abbildung 4 dargestellten Ergebnisse lassen sich für und die besten Ergebnisse erzielen. Für Split-MNIST variieren die besten Ergebnisse zwischen diesen beiden Werten von , abhängig vom Wert für . Bei ImageNet-10 können jedoch die besten Ergebnisse mit erzielt werden, weshalb dieser Wert im weiteren Verlauf der Arbeit genutzt wird.

Der Parameter ist maßgeblich dafür relevant, wie viele Repräsentationen angelegt werden. Damit beeinflusst direkt, wie groß der Speicherbedarf des FuzzyARTMAP-Netzwerks ist. Deshalb muss bei diesem Parameter auf einen Trade-Off zwischen Performanz und Speicherbedarf geachtet werden. Das erwartete Verhalten, bei dem ab einem gewissen Schwellwert kaum Veränderung in der Klassifikationsgenauigkeit beobachtet werden kann, kann durch die Versuche bestätigt werden. Bei Split-MNIST ist dieser Schwellwert bei ca. 0,9 zu sehen, während bei ImageNet-10 dieser bereits bei 0,7 zu beobachten ist. Dieser Unterschied kann mit der Komplexität der Bilder erklärt werden. Aufgrund der komplexen Bilder von ImageNet-10 sind die Ähnlichkeiten der einzelnen Bilder zueinander geringer, wodurch bereits bei niedrigeren Werten von nahezu jedes Trainingssample als Repräsentation angelegt wird. Bei Split-MNIST sind die besten Ergebnisse mit den Werten 0,9 und 1 zu beobachten. Das sind die Fälle, bei denen jedes Trainingssample als Repräsentation abgelegt wird. Bei ImageNet-10 sind die besten Ergebnisse für zu sehen.

Für Split-MNIST zeigt Abbildung 3 das erwartete Ergebnis für . Es wird die beste Klassifikationsgenauigkeit (87%) erreicht, da sehr viele Repräsentationen pro Klasse angelegt werden. Für ImageNet-10 ist dieses Verhalten nicht zu sehen. Dort werden die besten Ergebnisse (80-81%) für erreicht. Bei ist die Performanz wie erwartet stabil auf einem Niveau, allerdings nur bei ca. 77% Klassifikationsgenauigkeit. Ein Grund dafür könnte in den Bildern des ImageNet-Datensatzes liegen. Die Objekte sind in realer Umgebung zu sehen, und der Hintergrund/die Umgebung ist nicht schwarz wie bei MNIST. Dadurch kann es vorkommen, dass der Feature-Extrahierer (Modul A) unrelevante Features aus dem Hintergrund extrahiert (zum Beispiel starkes Vorkommen der Farbe Blau im Hintergrund bei Flugzeugen aufgrund des Himmels). Diese extrahierten Features geben jedoch keine Information über das Objekt. So können Features von Flugzeugen gleich (sehr ähnlich) denen von Vögeln sein, wenn beide im Himmel mit blauem Hintergrund abgebildet sind. Deshalb können sich einzelne, nicht generalisierte Repräsentationen den Validationsbildern von anderen Klassen ähneln, da einzelne Bilder der unterschiedlichen Klassen aufgrund nicht relevanter Features einen ähnlichen Feature-Vektor besitzen können.

Die genannten Schwellen spiegeln sich direkt im Speicherbedarf von Modul B wider. Ab dem genannten Schwellwert steigt der Speicherbedarf stark an, da viele Repräsentationen angelegt werden. Der mittlere Speicherbedarf von Modul B für Split-MNIST und ImageNet-10 ist in Abbildung 5 und Abbildung 6 graphisch dargestellt.

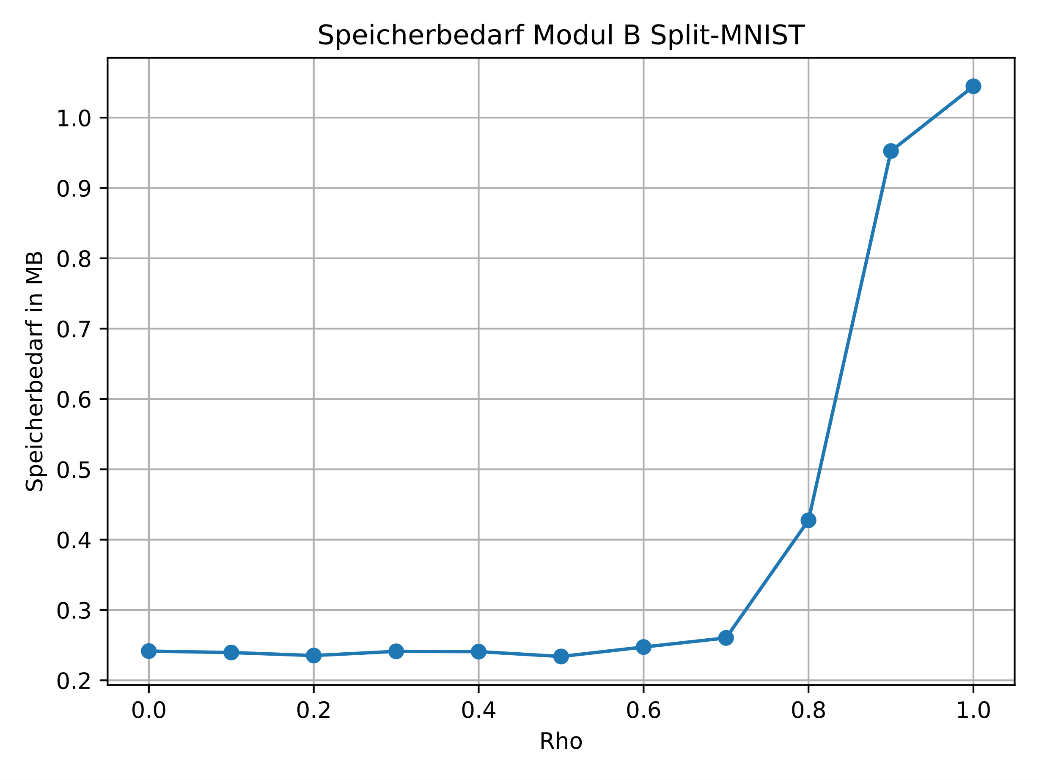


Abbildung 5: Speicherbedarf von Modul B in Abhängigkeit von für Split-MNIST

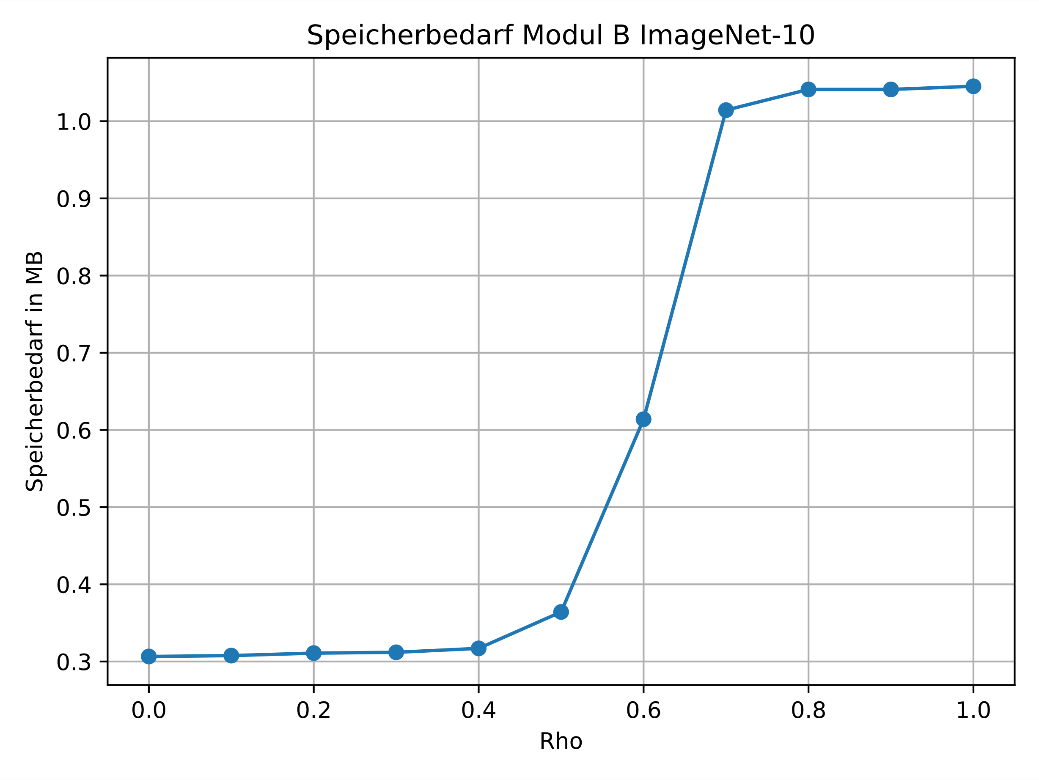


Abbildung 6: Speicherbedarf von Modul B in Abhängigkeit von für ImageNet-10

In diesen Abbildungen ist zu sehen, dass der Speicherbedarf ab dem Schwellwert (~0,9 für Split-MNIST und ~0,7 für ImageNet-10) auf das 3- bist 4-fache des davor benötigten Speicherbedarfs ansteigt.

Aufgrund der geringen Anzahl an Trainingsbildern (20 pro Klasse) ist der Speicherbedarf mit ca. 1MB in diesen Beispielen immer noch gering. Bei späteren Anwendungen mit mehr Klassen oder Trainingsbildern kann der Speicherbedarf jedoch schnell sehr groß werden. Auf Basis dieser Ergebnisse hinsichtlich Genauigkeit und Speicherbedarf wird für der Wert 0,5 gewählt. Mit diesem Parameterwert können auf beiden Datensätzen eine gute Klassifikationsgenauigkeit erzielt werden und der Speicherbedarf ist mit ca. 250 KB für Split-MNIST und ca. 350 KB für ImageNet-10 gering. Für weitere komplexere Anwendungen (z.B. gesamter ImageNet-Datensatz) wird der Speicherbedarf mit dieser Parametrierung in einem akzeptablen Bereich erwartet.

# Anzahl an Trainingsbilder pro Klasse

Schnell lernende inkrementelle Klassifikatoren sollen auf Basis von wenigen Beispieldaten einer Klasse diese erlernen können. Ein wichtiges Indiz ist dabei die Klassifikationsgenauigkeit in Abhängigkeit der gesehenen Anzahl an Samples pro Klasse. Damit kann untersucht werden, ab wie vielen Trainingssamples ein Maximum an Genauigkeit erreicht werden kann und wie sich der Performanz-Gewinn durch eine höhere Anzahl an Trainingsbilder verhält.

## Bezug zur Evaluierungsspezifikation

Wie in den Kriterien der Evaluierungsspezifikation dargestellt, wird dieser Test genutzt, um eine Aussage über die Fähigkeit des Algorithmus zu erhalten, schnell neue Klassen zu erlernen.

## Evaluierungsprozedur

Es werden die zuvor ermittelten Hyperparameter von Modul B verwendet. Mit diesen Parametern wird als weitere Untersuchung die Anzahl an Trainingsbildern variiert. Die Anzahl der Trainingsbilder wird dabei auf folgende Werte gesetzt: . Es werden pro Parameterwert 5 Wiederholungen durchgeführt, um einen aussagekräftigen statistischen Mittelwert und die Standardabweichung bilden zu können. Die weiteren einstellbaren Parameter sowie der Ablauf sind wie in Kapitel 2.2 beschrieben.

## Ergebnis

Es wird erwartet, dass mit mehr Trainingsbildern pro Klasse bessere Ergebnisse erzielt werden können, da mehr und besser generalisierte Repräsentationen für die einzelnen Klassen angelegt werden können. Mit steigender Anzahl an Trainingsbildern pro Klasse wird vermutet, dass die Varianz der Klassifikationsgenauigkeit abnimmt und damit stabilere Ergebnisse erzielt werden können. Die Annahme ist, dass bei wenigen Trainingsbildern die spätere Klassifikationsgenauigkeit stark von der Auswahl dieser wenigen Trainingsbilder abhängig ist. Bei steigender Anzahl an Trainingsbilder sinkt die Abhängigkeit von den einzelnen Trainingsbilder.

Zudem ist die Erwartung, dass der Speicherbedarf mit der Anzahl an Trainingsbildern steigt, da mehr Repräsentationen durch die höhere Zahl an gesehenen Trainingsdaten angelegt werden.

Die Ergebnisse für die Klassifikationsgenauigkeit werden in einem Balkendiagramm dargestellt. Die schwarzen Linien stellen dabei die Standardabweichung um den Mittelwert dar. Aufgrund der großen Unterschiede zwischen den einzelnen Werten für die Anzahl an Trainingsbilder wird die x-Achse in logarithmischer Skala dargestellt.

In Abbildung 7 ist die Klassifikationsgenauigkeit für Split-MNIST dargestellt.

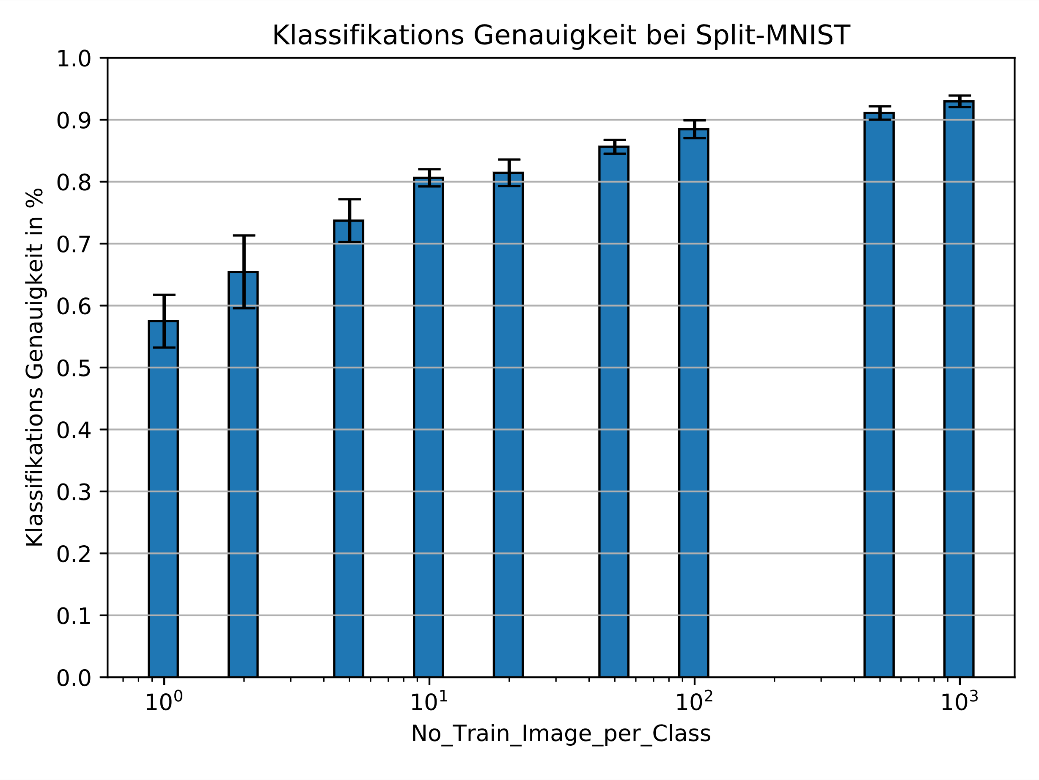


Abbildung 7: Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern Split-MNIST

Abbildung 8 stellt die Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern für ImageNet-10 dar.

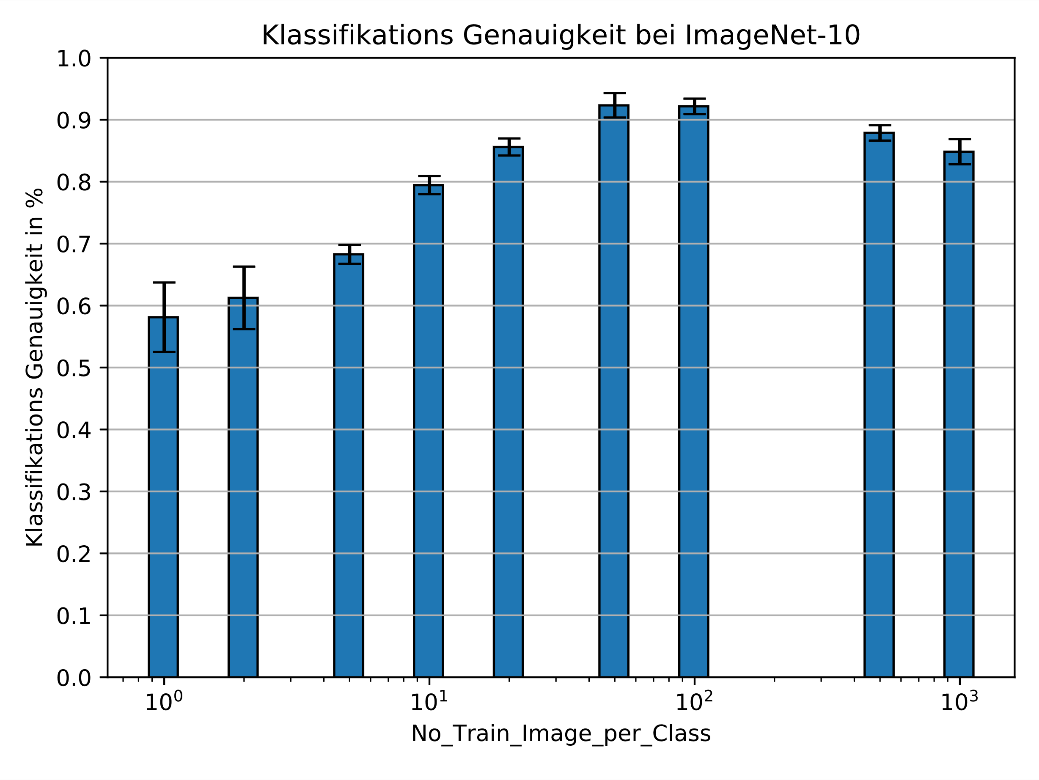


Abbildung 8: Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern ImageNet-10

Zusätzlich wird der mittlere Speicherbedarf von Modul B für die jeweiligen Datensätze in Abhängigkeit von der Anzahl an Trainingsbildern in Abbildung 9 für Split-MNIST und in Abbildung 10 für ImageNet-10 dargestellt.

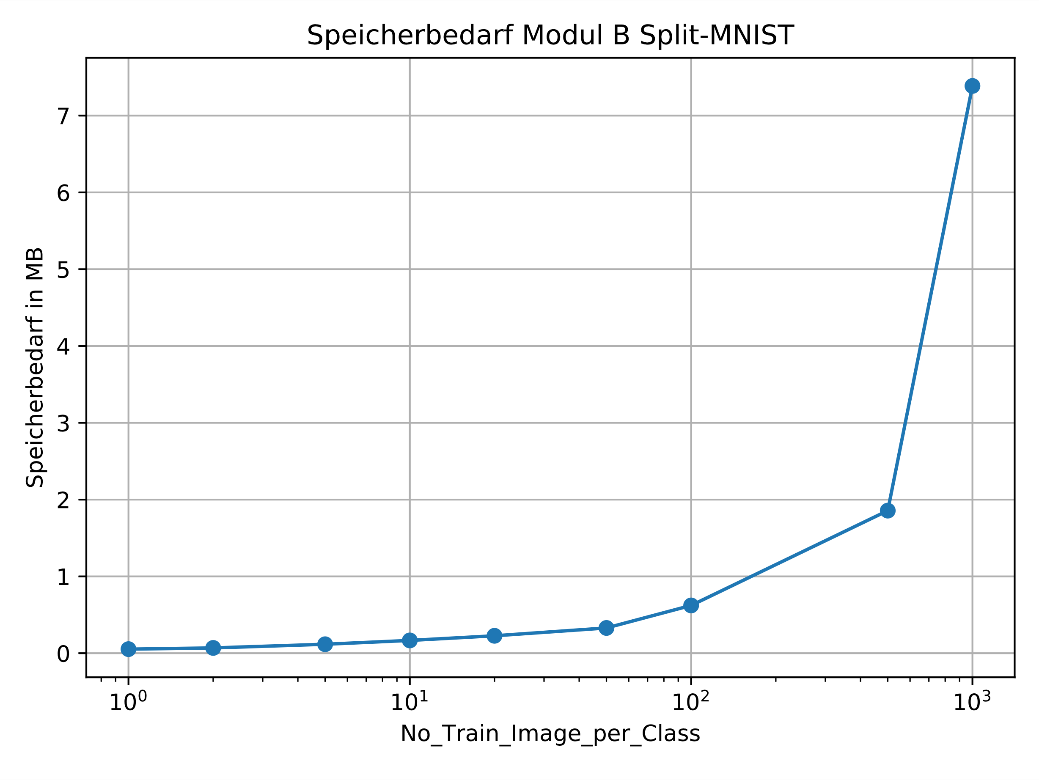


Abbildung 9: Speicherbedarf Modul B über die Anzahl an Trainingsbildern Split-MNIST

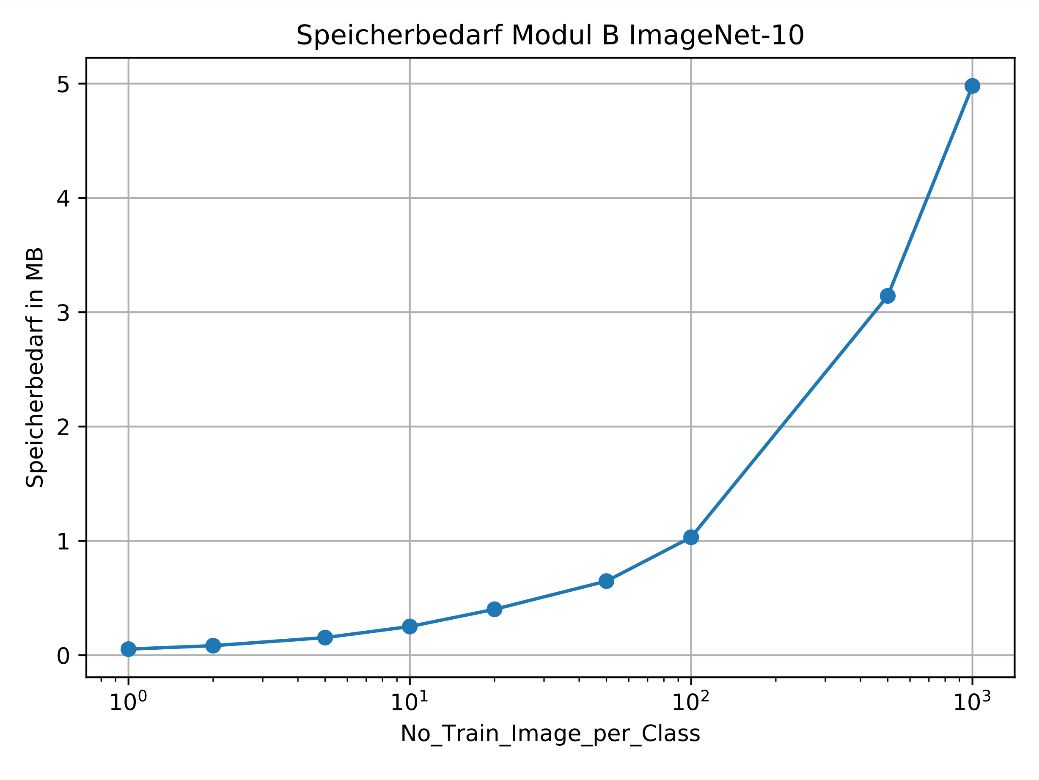


Abbildung 10: Speicherbedarf Modul B über die Anzahl an Trainingsbildern ImageNet-10

Die Auswertung der gezeigten Ergebnisse findet im folgenden Abschnitt 3.4 statt.

## Auswertung

Die Annahme, dass mit einer geringen Anzahl an Trainingsbildern eine hohe Varianz auftritt, kann auf Basis der erzielten Ergebnisse bestätigt werden. Für Split-MNIST (Abbildung 7) beträgt die Standardabweichung bei einem beziehungsweise zwei Trainingsbildern pro Klasse 4,3 und 5,9 Prozentpunkte. Mit 500 und 1000 Trainingsbildern pro Klasse beträgt die errechnete Standardabweichung lediglich 1,1 und 0,9 Prozentpunkte.

Bei ImageNet-10 ist ebenfalls eine höhere Standardabweichung bei einer geringen Anzahl Trainingsbildern zu sehen mit 5,6 und 5,1 Prozentpunkten bei einem und zwei Trainingsbildern pro Klasse. Die geringste Abweichung ist für ImageNet-10 mit jeweils 1,2 Prozentpunkten bei 100 und 500 Trainingsbildern pro Klasse zu finden.

Wenn die Klassifikationsgenauigkeit an sich betrachtet wird, kann für Split-MNIST gesagt werden, dass mit mehr Trainingsbildern pro Klasse die Genauigkeit besser wird. Bei 1000 Trainingsbildern pro Klasse wird im Mittel eine Klassifikationsgenauigkeit von 92,96% bei einer Standardabweichung von +/-0,94 erreicht. Mit 500 Trainingsbildern pro Klasse kann ebenfalls bereits eine gute Klassifikationsgenauigkeit von 91,08% +/- 1,1 erreicht werden.

Da aber zusätzlich der Speicherbedarf in Abbildung 9 in Betracht gezogen wird, werden im weiteren Verlauf der Arbeit 500 Trainingsbilder pro Klasse für Split-MNIST genutzt. Der Speicherbedarf steigt zwischen 500 und 1000 Trainingsbildern pro Klasse von ca. 1,9 MB auf ca. 7,2 MB an. Da Speicherbedarf in dieser Arbeit eine Rolle spielt, kann die leicht geringere Genauigkeit (ca. 1 Prozentpunkt) akzeptiert werden, wenn dafür weniger als ein Drittel an Speicher benötigt wird.

Für ImageNet-10 lässt sich ein anderes Verhalten der Klassifikationsgenauigkeit beobachten. Die Genauigkeit steigt zunächst bis 100 Trainingsbildern pro Klasse an, jedoch fällt die Genauigkeit im Gegensatz zu Split-MNIST daraufhin wieder ab. Ein möglicher Grund könnte der bereits beschriebene Fall von zu vielen Repräsentation sein (siehe die Auswertung des Parameters in Kapitel 2.4). Die beste Klassifikationsgenauigkeit lässt sich mit 50 und 100 Trainingsbildern pro Klasse erzielen, mit 92,31% +/- 1,9 und 92,16% +/- 1,2. Aufgrund der minimal geringeren Varianz und damit dem besseren „Worst-Case“ Ergebnis (90,41% vs. 90,96%) werden 100 Trainingsbilder pro Klasse für die weiteren Untersuchungen mit dem ImageNet-10 Datensatz genutzt. Auch der Speicherbedarf ist für 100 Trainingsbildern pro Klasse mit ca. 1 MB noch akzeptabel (Abbildung 10).

Insgesamt hat die Anzahl an Trainingsbildern pro Klasse einen, wie erwartet, direkten Einfluss auf die spätere Klassifikationsgenauigkeit. Bereits mit wenigen Trainingsbildern pro Klasse (z.B. 10) können jedoch akzeptable Klassifikationsgenauigkeiten von ca. 80% für die genutzten Validationsdaten von Split-MNIST und ImageNet-10 erreicht werden. Generell kann gesagt werden, dass mit steigender Anzahl an Trainingsbilder die Klassifikationsgenauigkeit ebenso wie der Speicherbedarf des inkrementellen Klassifikator zunimmt. Je nach Anwendungsfall kann eine geringere Genauigkeit infolge von weniger Trainingsdaten akzeptiert werden, wenn z.B. die Erzeugung von Trainingsdaten sehr kosten- und zeitaufwändig ist.

Mit diesem Testfall kann gezeigt werden, dass der hier untersuchte Algorithmus für spätere Anwendungen, bei denen wenigen Daten pro Klasse verfügbar sind, geeignet ist, da er bereits mit wenigen Trainingsbildern pro Klasse gute Ergebnisse erzielen kann.

# Untersuchung Continual Learning auf Testdaten

Für eine abschließende Bewertung und Einordnung des Potenzials des L DNN Algorithmus werden auf Basis der zuvor untersuchten Hyperparameter finale Tests für das kontinuierliche Lernen durchgeführt. Dabei wird eine größere Anzahl an Wiederholungen mit festen Parametern durchgeführt. Zudem werden die finalen Ergebnisse auf Basis der Testdaten der Datensätze ermittelt, nachdem zuvor für die Untersuchungen der Hyperparameter eine Kreuzvalidation mithilfe von Trainings- und Validationsdaten durchgeführt wurde.

## Bezug zur Evaluierungsspezifikation

Die in Kapitel 4 der Evaluierungsspezifikation genannten Testfälle werden nun durchgeführt. Für die Erstellung der folgenden Metriken werden Daten genutzt, die das Netzwerk bisher noch nicht gesehen hat. Dadurch kann eine Optimierung und *Overfitting* des Netzwerks auf Testdaten verhindert werden.

## Evaluierungsprozedur

Es werden die Testfälle des kontinuierlichen Lernens auf einem Gerät durchgeführt. Die Parameter werden auf Basis der vorherigen Ergebnisse ausgewählt und sind im Folgenden nochmals aufgelistet. Als Testdaten werden alle verfügbaren Testbilder der jeweiligen Datensätze genutzt.

* modul\_b\_epsilon = 0,001
* modul\_b\_s = 1,05
* modul\_b\_alpha = 0,2
* modul\_b\_rho = 0,5
* train\_img\_per\_class = 100 (ImageNet-10), 500 (Split-MNIST)
* test\_img\_per\_class = 50 (ImageNet-10), 1000 (Split-MNIST)

Es werden 10 Wiederholungen pro Datensatz durchgeführt.

## Ergebnis

In diesem Abschnitt werden die Ergebnisse dargestellt. Dafür wird für Split-MNIST eine Tabelle mit Ergebnissen anderer *Continual Learning*-Verfahren aus der Literatur sowie aktueller Ergebnisse mit traditionellen Deep Learning Ansätzen, hier einem Multy-Layer Perceptron (MLP) angelegt.

Für ImageNet-10 sind keine weiteren Ergebnisse von *Continual Learning*-Verfahren bekannt. Dieser Datensatz diente lediglich der Überprüfung des Potenzials auf komplexeren Eingangsdaten. Für einen Vergleich zu anderen Verfahren werden Test auf dem gesamten ImageNet-Datensatz durchgeführt (siehe Kapitel 7).

In Tabelle 1 sind die Ergebnisse für den Split-MNIST Datensatz dargestellt. Die ersten beiden Zeilen sind dabei der hier untersuchte Algorithmus. *L DNN Algorithmus inkrementell* ist der wie in der Evaluierungsprozedur beschriebene inkrementelle Algorithmus. *L DNN Algorithmus gesamt* ist dieselbe Architektur mit derselben Parametrierung, jedoch werden die Trainingsbilder aller Klassen gemeinsam in einem großen Batch trainiert und nicht inkrementell. Bei den weiteren Algorithmen ist angegeben, von welcher Quelle die genannte Klassifikationsgenauigkeit stammt. Die Ergebnisse gelten für den Fall des inkrementellen Klassen Lernens.

Tabelle 1: Klassifikationsgenauigkeit verschiedener Algorithmen auf Split-MNIST

|  |  |
| --- | --- |
| Algorithmus | Klassifikationsgenauigkeit in % |
| L DNN Algorithmus gesamt | 90,66 +/- 0,32 |
| L DNN Algorithmus inkrementell | 86,94 +/- 1,29 |
| Deep Generativ Replay (DGR) [2] | 91,24 +/- 0,33 |
| Elastic Weight Consolidation (EWC) [3] | 19,90 +/- 0,05 |
| Synaptic Intelligence (SI) [3] | 20,04 +/- 0,08 |
| Multi-Layer Perceptron (MLP) – inkrementell trainiert [3] | 19,90 +/- 0,02 |
| MLP – offline trainiert [3] | 97,93 +/- 0,04 |

EWC und SI stellen typische Methoden des kontinuierlichen Lernens dar. Diese Methoden speichern keine Trainingsdaten und nutzen keine gespeicherten Repräsentationen zum Training. Diese Abspeicherung und Verwendung der gespeicherten Daten im weiteren Trainingsverlauf wird in der Literatur *Replay* oder *Rehearsal* genannt. Deep Generative Replay (DGR) nutzt diese Methode, in dem es komprimierte Repräsentationen der Trainingsdaten abspeichert. Wenn neue Klassen hinzukommen, werden aus den gespeicherten Komprimierungen der alten Klassen sowie mithilfe eines erlernten generativen Moduls Trainingsbilder dieser Klassen erzeugt (Generative) und in die neuen Trainingsdaten eingebracht. Das *inkrementell trainierte MLP* kann als untere Grenze gesehen werden, da hier *Catastrophic Forgetting* aufgrund des *Backpropagation*-Algorithmus auftritt. Das *offline trainierte MLP* wurde mit allen Klassen offline trainiert, und kann als obere Grenze angesehen werden.

Für ImageNet-10 wird eine gemittelte Klassifikationsgenauigkeit von 76,4% +/-1,2 beim inkrementellen Erlernen erreicht. Vergleichbare Ergebnisse für diesen Anwendungsfall sind in der Literatur nicht zu finden. Als Referenz wird das Training der Architektur mit den Trainingsbildern aller Klassen herangezogen, wie für den *L DNN Algorithmus gesamt* bei Split-MNIST beschrieben. Mit diesem Training wird eine Klassifikationsgenauigkeit von 76,08% +/- 1,67 erreicht.

## Auswertung

Für Split-MNIST können die Ergebnisse mit anderen *Continual Learning* Algorithmen verglichen werden, da es viele Untersuchungen auf Basis dieses Datensatzes gibt. Im Vergleich zu den klassischen Methoden wie EWC und SI kann der L DNN Algorithmus deutlich bessere Ergebnisse für das hier untersuchte inkrementelle Klassen Lernen erzielen.

Algorithmen, die mit generativen Methoden arbeiten (wie DGR), erreichen eine bessere Klassifikationsgenauigkeit für diesen Anwendungsfall. Allerdings besitzen sie auch eine erhöhte Komplexität während des Trainings. Denn bei diesen Modellen muss zusätzlich zu dem inkrementellen Klassifikator ein generatives Modell trainiert werden, welches ausgewählte Trainingsdaten komprimiert und aus den komprimierten Darstellungen wiederherstellt (z.B. mithilfe eines *Variational Auto-Encoder*). Zudem müssen diese komprimierten Darstellungen abgespeichert werden. All diese Punkte ermöglichen generativen Modellen eine bessere Klassifikationsgenauigkeit, jedoch sind diese Modelle aus den genannten Gründen (noch) nicht für den Einsatz auf einem mobilen Endgerät geeignet, was in dieser Arbeit ein wichtiger Auswahlpunkt für den Algorithmus war. Insgesamt ist festzuhalten, dass eine Genauigkeit von ca. 87% auf Split-MNIST ein sehr gutes Resultat für das inkrementelle Klassenlernen ohne Replay/Rehearsal ist. Wenn lediglich Algorithmen ohne Rehearsal betrachtet werden, konnte in der Literatur keine bessere Klassifikationsgenauigkeit auf diesem Datensatz für das inkrementelle Klassenlernen gefunden werden. Als zusätzliche Referenz kann der *L DNN Algorithmus gesamt* gesehen werden. Dabei ist zu sehen, dass durch das inkrementelle Erlernen der Klassen ca. 3,5 Prozentpunkte Klassifikationsgenauigkeit verloren geht bei Split-MNIST.

Die Ergebnisse für ImageNet-10 können nicht mit anderen Algorithmen verglichen werden, jedoch kann damit geprüft werden, ob der Algorithmus auch auf komplexeren Eingangsdaten (64x64 RGB-Bilder mit komplexen Klassen) funktioniert, bevor ein großer und aufwändiger Test auf dem gesamten ImageNet-Datensatz durchgeführt wird. Mit einer finalen mittleren Klassifikationsgenauigkeit von 76,4% kann gesagt werden, dass der Algorithmus auch komplexere Klassen und Eingangsdaten korrekt klassifizieren und inkrementell erlernen kann. Auch wird beim inkrementellen Klassenlernen eine nahezu identische Genauigkeit (sogar minimal besser) wie beim Training der Architektur mit allen Trainingsbildern der Klassen (*L DNN Algorithmus gesamt*) erreicht.

# Untersuchung Distributed Learning auf Testdaten

Für eine abschließende Bewertung und Einordnung des Potenzials des L DNN Algorithmus werden auf Basis der zuvor untersuchten Hyperparameter finale Tests für das verteilte Lernen durchgeführt. Dabei wird eine größere Anzahl an Wiederholungen mit festen Parametern durchgeführt. Zudem werden die finalen Ergebnisse auf Basis der Testdaten der jeweiligen Datensätze ermittelt, nachdem zuvor für die Untersuchungen der Hyperparameter eine Kreuzvalidation mithilfe von Trainings- und Validationsdaten durchgeführt wurde.

## Bezug zur Evaluierungsspezifikation

Die in Kapitel 4 der Evaluierungsspezifikation genannten Testfälle für das verteilte Lernen werden nun durchgeführt. Für die Erstellung der folgenden Metriken werden Daten genutzt, die das Netzwerk bisher noch nicht gesehen hat. Dadurch kann eine Optimierung und *Overfitting* des Netzwerks auf Testdaten verhindert werden.

## Evaluierungsprozedur

Die Testfälle des verteilten Lernens werden auf zwei Geräten durchgeführt. Die Parameter werden auf Basis der vorherigen Testfälle ausgewählt und sind im Folgenden nochmals aufgelistet. Als Testdaten werden alle verfügbaren Testbilder der jeweiligen Datensätze genutzt.

* modul\_b\_epsilon = 0,001
* modul\_b\_s = 1,05
* modul\_b\_alpha = 0,2
* modul\_b\_rho = 0,5
* train\_img\_per\_class = 100 (ImageNet-10), 500 (Split-MNIST)
* test\_img\_per\_class = 50 (ImageNet-10), 1000 (Split-MNIST)

Es werden 10 Wiederholungen pro Datensatz durchgeführt.

## Ergebnis

Die Ergebnisse der einzelnen Datensätze werden hier dargestellt. Für den Fall des verteilten Lernens auf zwei Geräten wird die Klassifikationsgenauigkeit der einzelnen Geräte nach dem Erlernen ihrer verfügbaren Klassen angegeben. Zusätzlich wird die finale Genauigkeit des „verschmolzenen“ Netzwerks gegeben. Als Referenz dienen die Ergebnisse des kontinuierlichen Lernens auf einem Gerät, da im besten Fall durch das verteilte Lernen keine schlechteren Ergebnisse erzielt werden sollen.

Zunächst findet die Auswertung für Split-MNIST statt. Dabei wurden auf Gerät 1 die Gruppen 0/1, 2/3 und 4/5 trainiert, während auf Gerät 2 die Gruppen 6/7 und 8/9 trainiert wurden. Die gemittelten Ergebnisse aus 10 Läufen sowie deren Standard-Abweichung sind in Tabelle 2 gegeben.

Tabelle 2: Klassifikationsgenauigkeit des verteiltem L DNN Algorithmus auf Split-MNIST

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithmus | Klassifikations-genauigkeit Gerät 1 in % | Klassifikations-genauigkeit Gerät 2 in % | Klassifikations-genauigkeit final in % |
| L DNN Algorithmus – 1 Gerät | - | - | 86,94 +/- 1,29 |
| L DNN Algorithmus – 2 Geräte | 88,44 +/- 0,82 | 95,02 +/- 0,44 | 86,56 +/- 0,91 |

Dieselbe Evaluation wird auf Basis des ImageNet-10 Datensatzes durchgeführt. Die Ergebnisse dieser Untersuchung sind in Tabelle 3 zu sehen. Die beiden Geräte haben dieselbe Anzahl an Klassen (fünf) gesehen. Die jeweiligen Klassen, die auf den Geräten trainiert werden, sind ebenso wie die Reihenfolge der Klassen zufällig ausgewählt.

Tabelle 3: Klassifikationsgenauigkeit des verteiltem L DNN Algorithmus auf ImageNet-10

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithmus | Klassifikations-genauigkeit Gerät 1 in % | Klassifikations-genauigkeit Gerät 2 in % | Klassifikations-genauigkeit final in % |
| L DNN Algorithmus – 1 Gerät | - | - | 76,4 +/- 1,2 |
| L DNN Algorithmus – 2 Geräte | 84,64 +/- 4,73 | 87,44 +/- 4,04 | 76,26 +/- 1,5 |

Die Auswertung der Ergebnisse erfolgt im folgenden Kapitel 5.4.

## Auswertung

Die Ergebnisse für Split-MNIST und ImageNet-10 zeigen, dass mit separatem Training auf mehreren Geräten (hier zwei) dieselbe Genauigkeit erreicht werden kann wie mit dem Training auf einem einzelnen Gerät. Die Ergebnisse in Tabelle 2 und Tabelle 3 verdeutlichen diese Aussage. Zudem kann gesagt werden, dass durch das „Verschmelzen“ von Wissen eine schlechtere Genauigkeit auftritt. Die einzelnen Genauigkeiten von Gerät 1 und Gerät 2 auf ihren jeweiligen Testdaten sind besser als die finale Klassifikationsgenauigkeit. Dies hängt mit der komplexeren finalen Aufgabe zusammen, da dort statt 6 und 4 (für Split-MNIST) oder 5 und 5 Klassen (für ImageNet-10) nun 10 Klassen im Test-Datensatz vorkommen. Insgesamt kann auf Basis dieser Ergebnisse gesagt werden, dass das verteilte Lernen mit diesem Algorithmus funktioniert, da dort kein nennenswerter Performanz Verlust gegenüber dem zentralen Erlernen der Aufgabe auf einem Gerät auftritt. Somit eignet sich dieser Algorithmus zum Einsatz auf verteilten Systemen, die ihr Wissen ohne die jeweiligen Rohdaten austauschen müssen.

# Einfluss von Konsolidierungsschritten

Wie in der Konzeption beschrieben, gibt es die Möglichkeit das erlernte Wissen des inkrementellen Klassifikators zu konsolidieren. In diesem Prototyp wird die Konsolidierung durch eine Mittelwertbildung aller Repräsentationen einer einzelnen Klasse realisiert (für Details siehe Konzeption und Prototypenbeschreibung). Dadurch ist der Speicherbedarf um einiges geringer als ohne Konsolidierung. In diesem Testfall wird nun der genaue Einfluss der Konsolidierung auf die Klassifikationsgenauigkeit und den Speicherbedarf des inkrementellen Klassifikator in Modul B untersucht.

## Bezug zur Evaluierungsspezifikation

In der Evaluierungsspezifikation ist kein solcher Testfall spezifiziert. Er wurde im Verlauf der Arbeit spezifiziert und stellt eine erweiterte Untersuchung des Algorithmus dar.

## Evaluierungsprozedur

Es wird dieselbe Prozedur wie für die finalen Untersuchungen des kontinuierlichen Lernens genutzt (siehe Kapitel 4.2). Es wird zusätzlich eine Konsolidierung der Repräsentationen durchgeführt. Es werden zwei Testfälle durchgeführt. Beim ersten Testfall wird nach dem Training jeder Gruppe eine Konsolidierung durchgeführt. Dieser Testfall wird im weiteren Verlauf in Graphiken und Tabellen „*Konsolidierung jeder Schritt*“ genannt. Im zweiten Testfall findet keine Konsolidierung während des Trainings der einzelnen Gruppen statt. Die Konsolidierung findet nach dem Training aller Gruppen/Klassen und vor der Bestimmung der Test-Genauigkeit statt. Dieser Fall wird im weiteren Verlauf „*Konsolidierung finaler Schritt*“ genannt. Als Referenz dienen die in Kapitel 4 erzielten Ergebnisse ohne Konsolidierung. Für die beiden Testfälle werden jeweils 10 Wiederholungen durchgeführt. Die im Folgenden genannten Ergebnisse beziehen sich dabei jeweils auf die Mittelwerte der erhaltenen Ergebnisse. Die Tests werden für Split-MNIST und ImageNet-10 durchgeführt.

## Ergebnis

Zunächst wird die Klassifikationsgenauigkeit der unterschiedlichen Testfälle auf Basis von Split-MNIST dargestellt. Die Ergebnisse für die beiden Testfälle sowie der Referenz (Keine Konsolidierung) sind in Abbildung 11 zu sehen. Die unterschiedlichen Kurven stellen dabei die Genauigkeiten für die jeweiligen Methoden der Konsolidierung über die Anzahl an erlernten Klassen dar.

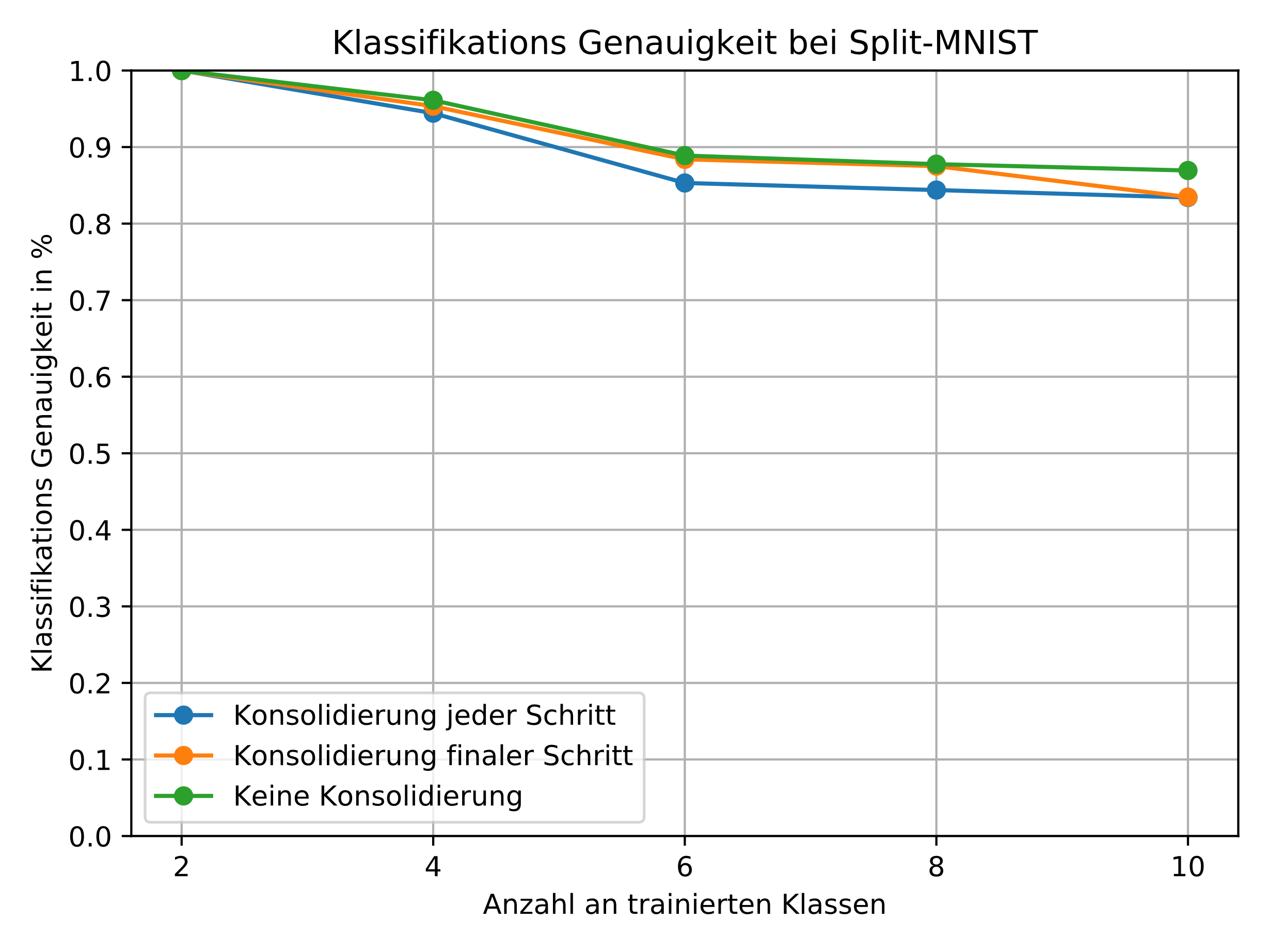


Abbildung 11: Klassifikationsgenauigkeit für unterschiedliche Konsolidierungsmethoden

Die gleiche Darstellung der Ergebnisse für ImageNet-10 ist in Abbildung 12 zu sehen.

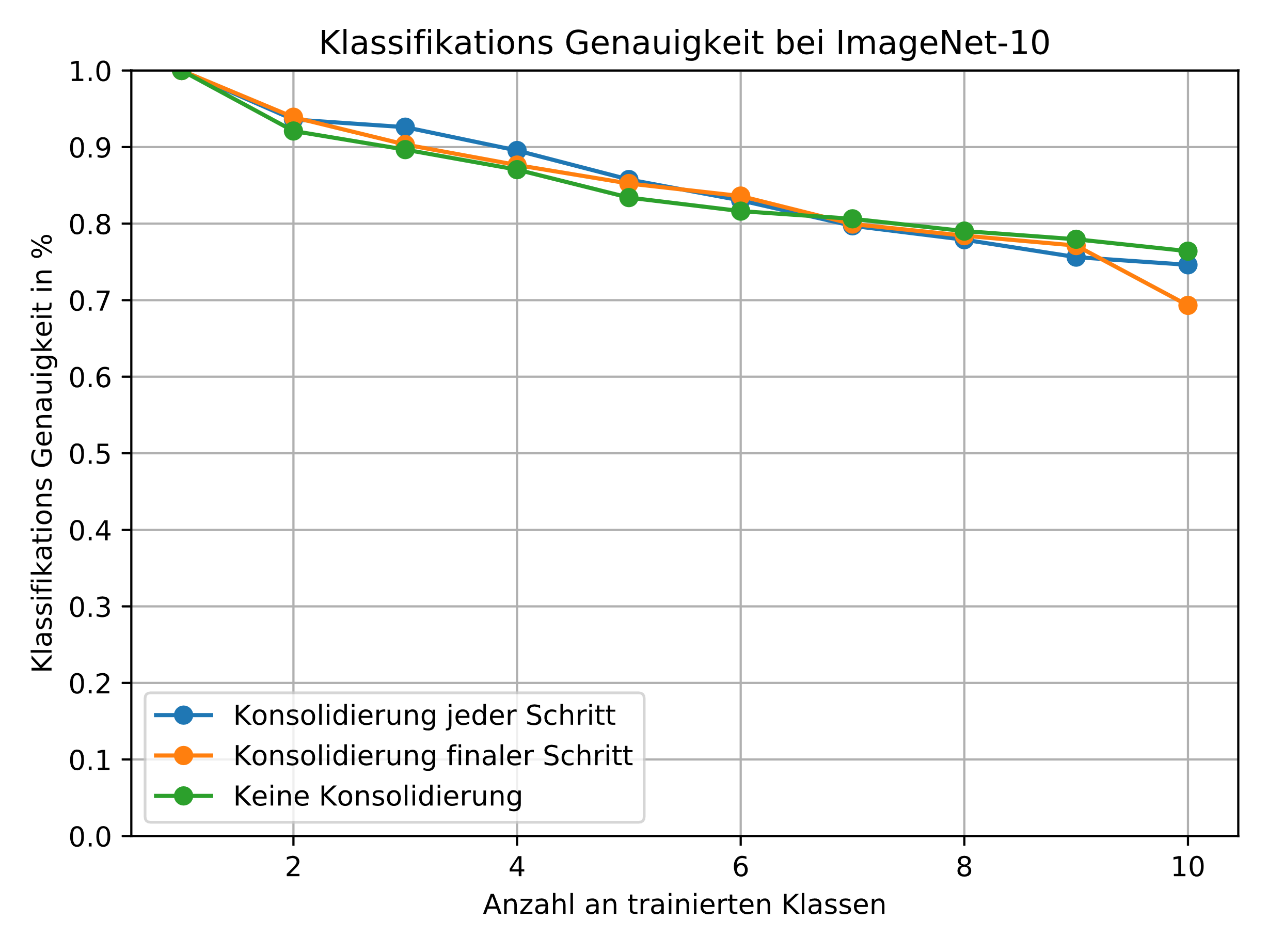


Abbildung 12: Klassifikationsgenauigkeit für unterschiedliche Konsolidierungsmethoden

Für eine sinnvolle Einschätzung und Bewertung wird der finale Speicherbedarf der unterschiedlichen Methoden angeschaut. Dieser wird inklusive der finalen Klassifikationsgenauigkeiten für Split-MNIST in Tabelle 4 zusammengefasst.

Tabelle 4: Vergleich von Speicherbedarf und finaler Klassifikationsgenauigkeit für unterschiedliche Methoden der Konsolidierung Split-MNIST

|  |  |  |
| --- | --- | --- |
| Konsolidierungs-methode | Finale Klassifikations-genauigkeit Split-MNIST in % | Finaler Speicherbedarf Split-MNIST in MB |
| Keine Konsolidierung | 86,94 +/-1,29 | 1,83 |
| Jeder Schritt | 83,41 +/- 1,64 | 0,1 |
| Finaler Schritt | 83,47 +/- 1,8 | 0,1 |

Tabelle 5 zeigt diesen Zusammenhang für ImageNet-10.

Tabelle 5: Vergleich von Speicherbedarf und finaler Klassifikationsgenauigkeit für unterschiedliche Methoden der Konsolidierung ImageNet-10

|  |  |  |
| --- | --- | --- |
| Konsolidierungs-methode | Finale Klassifikations-genauigkeit ImageNet-10 in % | Finaler Speicherbedarf ImageNet-10 in MB |
| Keine Konsolidierung | 76,4 +/- 1,2 | 0,96 |
| Jeder Schritt | 74,62 +/- 2,19 | 0,1 |
| Finaler Schritt | 69,32 +/- 4,81 | 0,1 |

## Auswertung

Anhand der erzielten Ergebnisse kann gesagt werden, dass bei Anwendungen mit sehr begrenztem Speicher die Konsolidierung eine gute Maßnahme sein kann, um den Speicherbedarf drastisch zu reduzieren und dabei geringe Genauigkeitseinbußen zu haben. Für Split-MNIST wird eine finale Klassifikationsgenauigkeit von ca. 83% mit beiden Konsolidierungsmethoden erreicht, was ca. 3 Prozentpunkte schlechter ist als ohne Konsolidierung. Jedoch ist der Speicherbedarf mit 0,1 MB auch nur 1/18 im Vergleich zu keiner Konsolidierung (siehe Tabelle 4). In Abbildung 11 kann gezeigt werden, dass bei einer Konsolidierung im finalen Schritt die Genauigkeit bis zum letzten finalen Testlauf identisch bleibt, da dort noch keine Konsolidierung stattgefunden hat und somit das Training und Testen bis dorthin identisch ist. Für Split-MNIST kann kein direkter Unterschied zwischen den beiden Konsolidierungsmethoden („*Jeder Schritt*“ und „*Finaler Schritt*“) gesehen werden, da beide nahezu identische Ergebnisse liefern.

Diese Aussage gilt nicht für ImageNet-10. Auch hier ist der Speicherbedarf mit 0,1 MB im Vergleich zu 0,96 MB um einiges geringer. Bei der Klassifikationsgenauigkeit gibt es jedoch auffällige Unterschiede. Mit der Konsolidierung nach jedem Schritt kann mit 74,62% Genauigkeit ein ähnliches Resultat wie ohne Konsolidierung (76,4%) erreicht werden. Bei der Konsolidierung nach dem finalen Schritt wird jedoch eine Genauigkeit von lediglich 69,32% mit deutlich erhöhter Varianz (4,81) erzielt (siehe Tabelle 5). Dies verdeutlicht den Einfluss der Konsolidierungsmethode für ein FuzzyARTMAP-Netzwerk, da das Training eines solchen Netzwerks stark von bereits bekannten Repräsentationen abhängt.

Der Grund für die schlechtere Performanz der Methode „finaler Schritt“ könnte darin liegen, dass während der Trainingsschritte viele Repräsentationen anderer Klassen vorliegen. Dies führt in der Regel zu einer erhöhten Anzahl an Repräsentationen für neue Klassen, um einen eindeutigen Sieger für diese Kategorie stellen zu können. Für einen Datensatz mit komplexen Klassen und Bilder wie ImageNet wird dann bei einer Konsolidierung im finalen Schritt separates Wissen über eine Klasse/Kategorie mit einer simplen Mittelwert-Logik verbunden. Dadurch werden unterschiedliche Repräsentationen „in ihrer Mitte“ zusammengefasst. Wenn die Methode „*Jeder Schritt*“ angewendet wird, ist während des Trainings einer neuen Klasse nur eine Repräsentation pro alte Klasse vorhanden. Dadurch bildet das FuzzyARTMAP-Netzwerk weniger Repräsentationen mithilfe der optimierten Lernrate . Diese generalisierten Repräsentationen scheinen robuster zu sein als die mithilfe des Mittelwertes konsolidierten Repräsentationen.

Alles in allem kann mithilfe der Konsolidierung der Speicherbedarf drastisch reduziert. In Abhängigkeit der gewählten Konsolidierungsmethode wird dabei die Klassifikationsgenauigkeit nur leicht verringert. Mit der Methode „*Jeder Schritt*“ können ähnliche Ergebnisse wie ohne Konsolidierung erreicht werden mit 1/18 (Split-MNIST) und 1/9 (ImageNet-10) des ursprünglichen Speicherbedarfs. Dies zeigt, dass Konsolidierung für eine spätere Anwendung in einem speicherarmen Endgerät interessant sein kann.

Als Ausblick könnte für die Konsolidierung eine mathematisch komplexere Formel anstatt dem simplen Mittelwert genutzt werden. Zum Beispiel könnten nur Repräsentationen einer Klasse konsolidiert werden, die eine Ähnlichkeit (z.B. *cosine-*Similarity) über einem definierten Schwellwert haben. Dadurch kann das konsolidieren von im Feature-Raum unterschiedlichen Repräsentationen einer Klasse vermieden werden. Als Folge würden die Klassen nicht durch eine, sondern durch mehrere ausgewählten Repräsentationen dargestellt, die wiederum eine Vielzahl an ähnlichen Repräsentationen bündeln und darstellen. Dies kann im Rahmen dieser Arbeit jedoch aufgrund von Zeitgründen nicht weiterverfolgt werden.

Als weiteren Ausblick kann mit der hier vorhandenen Logik ein intelligenter Zeitpunkt für die Konsolidierung gewählt werden. Es muss nicht nach dem Training einer neuen Klasse stattfinden, wenn dafür keine Notwendigkeit herrscht, sondern könnte in realen Anwendungsfällen zum Beispiel zu dem Zeitpunkt erfolgen, an dem 90% des verfügbaren Speichers belegt ist. Wie in den hier beschriebenen Testfällen gezeigt, kann bei geringem Genauigkeitsverlust (2-3 Prozentpunkte) eine Menge Speicher freigegeben werden (Reduktion auf 1/18 und 1/9 des ursprünglichen Bedarfs).

# ImageNet

Für eine finale Untersuchung des Algorithmus auf einem komplexen Datensatz mit vielen Klassen wird der ImageNet-2012 Datensatz genutzt. Dieser beinhaltet 1000 Klassen mit über 1 Millionen Trainingsbildern und 50.000 Testbilder. Die Ergebnisse auf diesem Datensatz dienen als Vergleich zu anderen inkrementellen Klassifikatoren. Der Datensatz ist detaillierter in der Evaluierungsspezifikation und in Kapitel 1 dieses Dokuments definiert. Es wird zunächst das inkrementelle Klassenlernen auf einem Gerät untersucht, und später eine Untersuchung des Lernverhaltens auf zwei verteilten Geräten durchgeführt.

## Bezug zur Evaluierungsspezifikation

Die in Kapitel 4 der Evaluierungsspezifikation genannten Testfälle für ImageNet werden nun durchgeführt. Für eine Vergleichbarkeit mit anderen Algorithmen wird das Netzwerk nicht wie in der Evaluierungsspezifikation beschrieben zunächst auf 900 zufälligen Klassen vortrainiert, sondern er wird inkrementell mit allen 1000 Klassen trainiert. Für den Fall des verteilten Lernens auf zwei Geräten werden jeweils 500 Klassen pro Gerät trainiert.

## Evaluierungsprozedur

Es werden die in Kapitel 4.2 und 5.2 genannten Parameterwerte für ImageNet-10 auch für den gesamten ImageNet Datensatz genutzt. Variiert wird die Anzahl an inkrementellen Schritten (Trainings-Gruppen). Dies verändert die Anzahl an Klassen, die pro Trainingsschritt erlernt werden. Es werden dabei folgende Werte für die Anzahl an Klassen pro Trainingsschritt gewählt: [1, 50, 100, 1000]. Pro Fall wird aufgrund der großen Datenmenge anstatt wie bisher 10 lediglich 1 Wiederholung durchgeführt. Zudem werden aufgrund der großen Anzahl an Klassen lediglich 10 Trainingsbilder pro Klasse genutzt anstatt 100 wie für ImageNet-10. Als Referenz werden die in [4] aufgeführten Ergebnisse für ImageNet genutzt.

## Ergebnis

Zunächst wurde der Algorithmus mit 1 inkrementellem Schritt trainiert. Das heißt, dass kein inkrementelles Lernen stattfindet, sondern alle 1000 Klassen im ersten Trainingsschritt erlernt werden. Es konnte eine finale Klassifikationsgenauigkeit auf den Testdaten von 39,1% erreicht werden bei einem Speicherbedarf von 85,2 MB für Modul B. Da hier kein inkrementelles Lernen stattfindet, wird auf eine graphische Darstellung dieses einen Wertes verzichtet.

Mit 10 inkrementellen Schritten wird am Ende auf allen Testdaten eine Klassifikationsgenauigkeit von 39,3% bei einem Speicherbedarf von 87,2 MB erreicht. Da diese Anzahl an inkrementellen Schritten auch in [4] für die finale Auswertung genutzt wird, werden die erreichten Ergebnisse in Relation zu dem bereits in der Konzeption beschriebenen *iCaRL*-Algorithmus sowie dem *Learning without Forgetting* (LwF) [5] gesetzt. Die Ergebnisse sind in Abbildung 13 dargestellt.

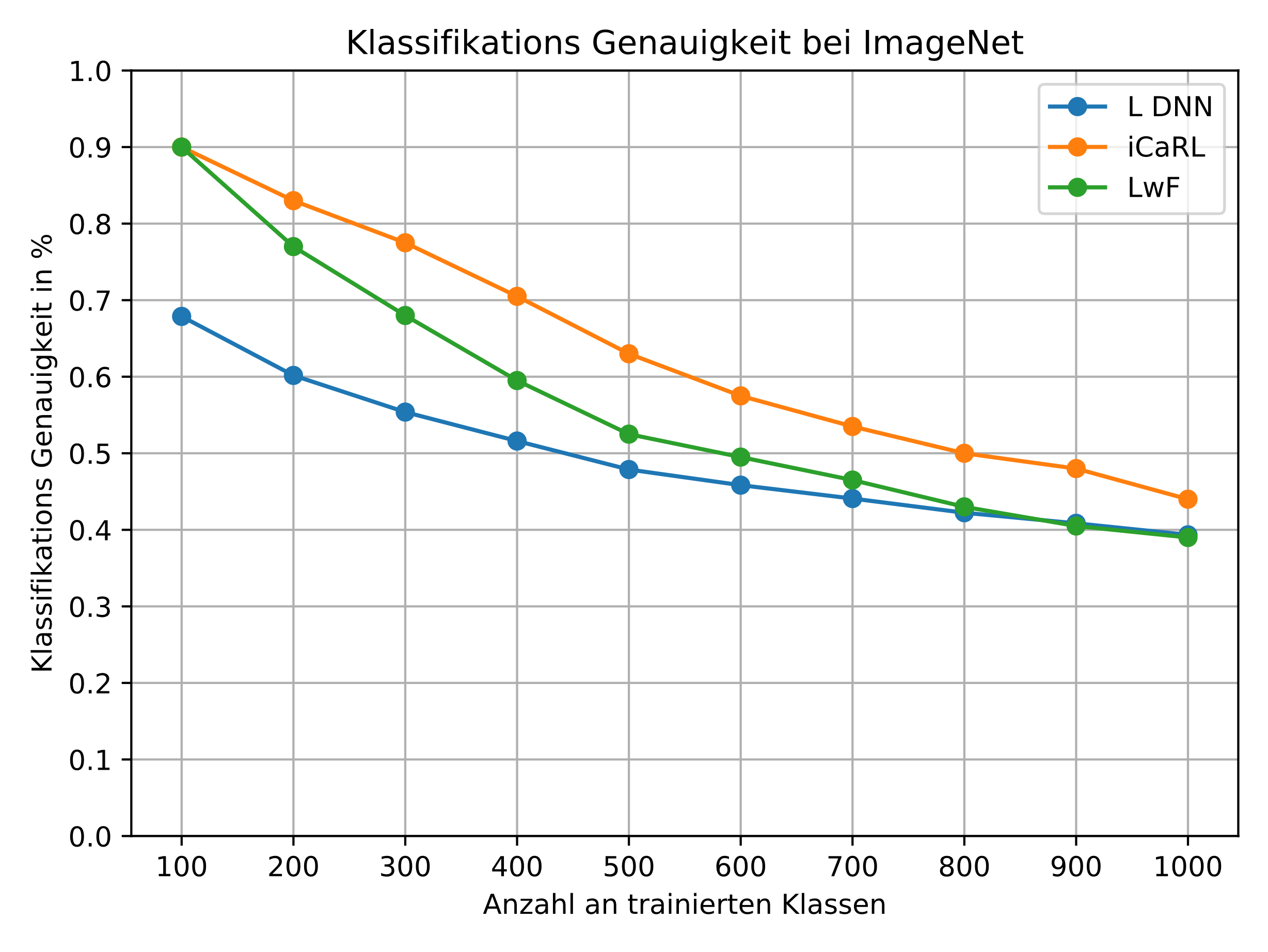


Abbildung 13: Klassifikationsgenauigkeit bei ImageNet für unterschiedliche inkrementelle Lernalgorithmen

Zusätzlich wird in Abbildung 14 zu den bereits beschriebenen Kurven die Klassifikationsgenauigkeit des L DNN Algorithmus mit 20 inkrementellen Schritten dargestellt (rote Kurve). Die anderen Kurven sind identisch zu Abbildung 13 und nur als Referenz eingezeichnet.

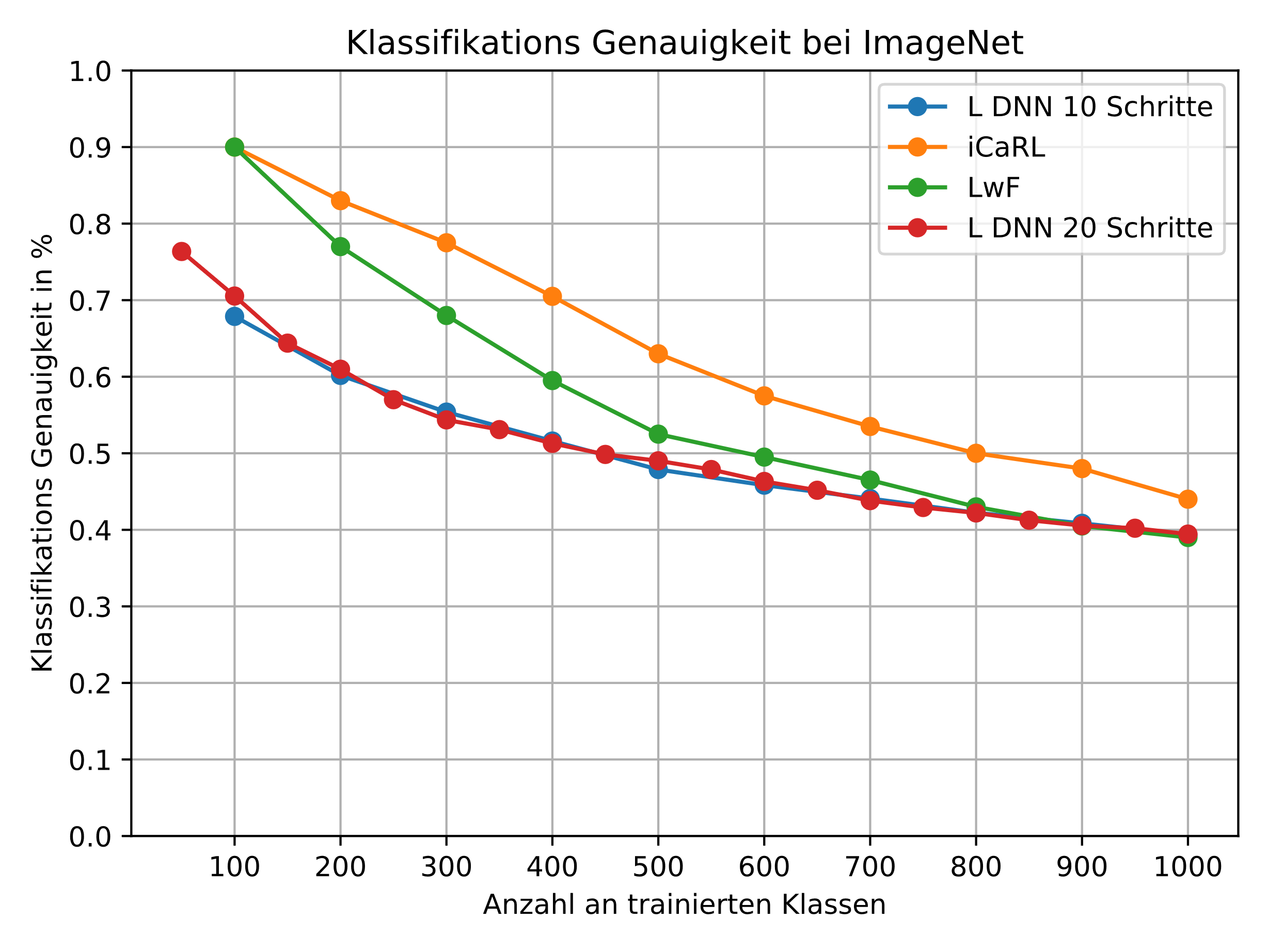


Abbildung 14: Klassifikationsgenauigkeit bei ImageNet mit unterschiedlicher Anzahl an inkrementellen Schritten

Beide Algorithmen, iCaRL und LwF, nutzen ein 18-Layer ResNet zur Feature-Extrahierung. Der iCaRL-Algorithmus speichert 20.000 Exemplare in dem referenzierten Versuch ab. LwF arbeitet ohne Speicherung von Exemplaren und trainiert neue unabhängige Fully Connected Layer für jeden inkrementellen Schritt. Jeder inkrementelle Lernschritt hat bei diesen beiden Algorithmen 100 Epochen, während der L DNN Algorithmus lediglich eine Epoche pro inkrementellen Schritt durchführt. In Tabelle 6 sind die finalen Genauigkeiten und der Speicherbedarf der genannten Algorithmen zusammengefasst.

Tabelle 6: Finale Klassifikationsgenauigkeiten ImageNet

|  |  |  |
| --- | --- | --- |
| Algorithmus | Finale Klassifikations-genauigkeit ImageNet in % | Finaler Speicherbedarf ImageNet-10 in MB |
| L DNN  (ein inkrementeller Schritt) | 39,1 | 527,2 |
| L DNN  (10 inkrementelle Schritte) | 39,3 | 530,5 |
| L DNN  (50 inkrementelle Schritte) | 39,4 | 528,8 |
| iCaRL | 44 | 2123 |
| LwF | 39 | - |

Für L DNN ist der gesamte Speicherbedarf für Modul A und Modul B angegeben. Für iCaRL und LwF konnten keine konkreten Angaben zum Speicherbedarf gefunden werden. Für iCaRL wurde der Speicherbedarf mithilfe der durchschnittlichen Größe eines Bildes (ca. 106 KB) des ImageNet-Datensatz ermittelt für 20.000 gespeicherte Exemplare. Zusätzlich würde noch der Speicherbedarf des Feature-Extrahierers dazu kommen. Für LwF konnte keine Angabe für den Speicherbedarf gefunden werden und auch keine sinnvolle Abschätzung getroffen werden.

## Auswertung

Auf Basis der Ergebnisse, die in Tabelle 6 und in Abbildung 13 dargestellt sind, wird der L DNN Algorithmus final bewertet. Dafür wird zunächst der L DNN Algorithmus mit 10 inkrementellen Schritten mit iCaRL und LwF verglichen. In Abbildung 13 ist zu sehen, dass der L DNN Algorithmus vom ersten inkrementellen Schritt (100 Klassen) an eine deutlich geringe Klassifikationsgenauigkeit hat (67,9%) als iCaRL und LwF (beide 90%). Dies kann mit dem gewählten Feature-Extrahierer begründet werden. Mit MobileNet-v2 wurde für den L DNN Algorithmus ein Kompromiss zwischen Speicherbedarf und Genauigkeit getroffen. LwF und iCaRL nutzen jeweils eine ResNet-Architektur für die Feature-Extraktion. Diese Architektur erzielt auf ImageNet bessere Genauigkeiten als MobileNet-v2 (siehe Konzeption), wodurch eine höhere Klassifikationsgenauigkeit mit den hier genutzten Algorithmen möglich ist. Ebenfalls ist in Abbildung 13 der Verlauf über die Anzahl an trainierten Klassen zu sehen. Der L DNN Algorithmus ist hier wesentlich stabiler als die anderen beiden Algorithmen, welche mit zunehmender Anzahl an trainierten Klassen deutliche stärker an Genauigkeit verlieren. Tabelle 7 stellt den relativen verbleibenden Anteil der Klassifikationsgenauigkeit für die einzelnen Algorithmen dar. Dafür wird die Genauigkeit nach dem letzten finalen Schritt (1000 Klassen) durch die Genauigkeit nach dem ersten Schritt (100 Klassen) dividiert, um das relative Verhältnis zu erhalten.

Tabelle 7: Relativer Erhalt der Klassifikationsgenauigkeit auf ImageNet

|  |  |
| --- | --- |
| Algorithmus | Relativer Erhalt Klassifikationsgenauigkeit in % |
| L DNN  (10 inkrementelle Schritte) | 57,9 |
| iCaRL | 48,9 |
| LwF | 43,3 |

Der L DNN Algorithmus erreicht nach dem finalen Training auf 1000 Klassen noch ca. 58% seiner anfänglichen Genauigkeit, während iCaRL und LwF mit ca. 49% (iCaRL) und ca. 43% (LwF) jeweils weniger als die Hälfte erreichen.

Auf Basis dieser Ergebnisse kann gesagt werden, dass der L DNN Algorithmus für eine große Anzahl an Klassen stabiler ist und im Vergleich zu den anderen Algorithmen auf großen Daten „besser skaliert“.

Die finale Genauigkeit des L DNN Algorithmus ist dabei identisch wie die des LwF-Algorithmus, der jedoch eine komplexere Trainingsstrategie verfolgt (Training einzelner Fully-Connected Layer mithilfe des Backpropagation-Algorithmus auf Basis der neuen Samples). Hier ist kein Vergleich beim Speicherbedarf möglich, da für LwF keine Werte vorliegen und der Speicherbedarf nicht abgeschätzt werden kann. Der iCaRL-Algorithmus erreicht eine bessere finale Klassifikationsgenauigkeit mit 44% gegenüber dem L DNN Algorithmus (ca. 39%), jedoch ist der Speicherbedarf hier um einiges höher. Für die Speicherung der Exemplare benötigt iCaRL bei diesem Test über 2 GB, während der gesamte L DNN Algorithmus lediglich ca. 530 MB benötigt. Unter Berücksichtigung der genutzten Feature-Extrahierer ist es möglich, dass der L DNN Algorithmus mit einem leistungsfähigeren Extrahierer eine bessere Genauigkeit als iCaRL erreichen kann. Der Speicherbedarf des L DNN würde steigen, jedoch vermutlich weiterhin geringer sein als der des iCaRL. Der gesteigerte Speicherbedarf könnte in einzelnen Anwendungsfällen für eine bessere Genauigkeit akzeptiert werden.

Im Weiteren wird der Einfluss von einer unterschiedlichen Anzahl an inkrementellen Schritten für den L DNN Algorithmus bei einem großen Datensatz untersucht. Hierzu wurden 20 statt 10 inkrementelle Schritte durchgeführt. Damit werden bei einem inkrementellen Trainingsschritt 50 zufällige Klassen trainiert. Auf Basis der in [4] angegebenen Resultate führt eine erhöhte Anzahl an inkrementellen Schritten bei LwF und iCaRL zu einer schlechteren finalen Klassifikationsgenauigkeit. Speziell LwF ist deutlich sensitiver zu der Anzahl an inkrementellen Trainingsschritten. Dieses Verhalten ist für den L DNN Algorithmus nicht zu beobachten. In Abbildung 14 kann kein nennenswerter Unterschied zwischen den Kurven mit 10 und 20 inkrementellen Schritten gesehen werden. Auch die finale Klassifikationsgenauigkeit ist (nahezu) identisch. Dieses Verhalten ist für spätere reale Anwendungen von großer Bedeutung. In relevanten Anwendungen des kontinuierlichen Lernens können viele inkrementelle Schritte auftreten, da einzelne Klassen nach und nach während dem Betrieb auftreten. Dort soll eine gute Klassifikationsgenauigkeit ermöglicht werden. In diesem Fall ist für LwF und iCarL eine deutlich geringere Performanz im Vergleich zu den bekannten Ergebnissen auf den hier genutzten Test-Datensätzen zu erwarten. Für den L DNN Algorithmus kann erwartet werden, dass die hier gezeigte und erreichte Performanz auch bei vielen inkrementellen Schritten erreicht werden kann.

Dieses hier beobachtete Verhalten des L DNN Algorithmus ist positiv für die Nutzung des L DNN Algorithmus in weiteren (realen) Anwendungen. Der Algorithmus erfüllt die grundlegenden Eigenschaften eines kontinuierlich lernenden Algorithmus, und kann auf Basis weniger Trainingsdaten robust neue Klassen erlernen. Zudem verhält er sich gegenüber einer großen Anzahl an Klassen stabil und ist ebenfalls nicht sensitiv gegenüber der Anzahl an inkrementellen Trainingsschritten.

Es sollte weiterhin untersucht werden, inwieweit die Klassifikationsgenauigkeit des Algorithmus durch einen besseren Feature-Extrahierer in Modul A verbessert werden kann. Wenn hier eine weitere Verbesserung der Genauigkeit erreicht werden kann, hat der Algorithmus das Potenzial auch in realen Anwendungen sehr gute Resultate zu erzielen. Zudem wäre eine Untersuchung auf ImageNet mit 1000 inkrementellen Schritten (1 Klasse pro Schritt) interessant für die im vorigen Abschnitt getroffene Aussage. Weiterhin sollte der Verhalten des verteilten Lernens für 2 (oder mehrere) Endgeräte auf ImageNet untersucht werden.

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | 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,“ 30 Januar 2015. |
| [2] | Y.-c. Hsu, Y.-c. Liu und Z. Kira, „Re-evaluating Continual Learning Scenarios : A Categorization and Case for Strong Baselines,“ in *32nd Conference on Neural Information Processing Systems (NIPS2018)*, Montréal, 2018. |
| [3] | G. M. van de Ven und A. S. Tolias, „Three continual learning scenarios and a case for generative replay,“ in *International Conference on Learning Representations*, New Orleans, 2019. |
| [4] | Y. Wu, C. Yinpeng, L. Wang, Y. Ye, Z. Liu, G. Yandong und Y. Fu, „Large Scale Incremental Learning,“ *CoRR,* Mai 2019. |
| [5] | Z. Li und D. Hoiem, „Learning without Forgetting,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 2935-2947, 2018. |