|  |  |  |
| --- | --- | --- |
| **MA 3062** | | |
| Untersuchung und prototypische Umsetzung eines Lifelong Deep Neural Network Algorithmus | | |
| **Simon Kamm** | | |
|  | | |
|  | | |
|  | 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 |
|  |  |  |  |  |  |

*Allgemeine Hinweise zur Benutzung von IAS-VM Dokumentvorlagen*

*Diese Hinweise und Erläuterungen in den einzelnen Kapiteln der Vorlage sind kursiv dargestellt und müssen im Dokument gelöscht werden.*

*Verwendete Symbole:*

*\* generisch (kann mehrfach vorkommen)*

*[...] optional (kann auch weggelassen werden)*

*<...> Platzhalter (Bereich inklusive Klammern muss entsprechend ersetzt werden)*

***Bitte ändern Sie die Formatierung des Deckblatts nicht sondern tragen Sie nur die abgefragten Daten ein!***

*Bei der Erstellung eines Dokumentes hat das Dokument zunächst den Status ‘in Bearb.’ und die Versionsnummer 0.1. Dokumente im Status ‘in Bearb.’ können beliebig geändert, gespeichert und gedruckt werden. Wichtige Änderungen sollten allerdings im Feld „Änderungen“ dokumentiert werden. Die erste fertige Version bekommt die Versionsnummer 1.0. Ab der Version 1.0 wird das Speichern im Zustand ‘vorgelegt’ erlaubt. Dokumente mit niedrigerer Versionsnummer dürfen nicht mit ‘vorgelegt’ gespeichert werden.* ***Diese Informationen müssen manuell eingetragen werden.***

*Dokumente unterliegen ab dem Status ‘vorgelegt’ dem Konfigurationsmanagement und dürfen nicht mehr überschrieben werden. Nach einer Änderung muss das Dokument unter Angabe der durchgeführten Änderungen mit neuer Versionsnummer gespeichert werden. Die Versionsnummern muss dabei um 0.1 hochgezählt werden.*

*Bei der Erstellung eines Dokuments ist folgendes zu beachten:*

* *Wurden alle Evaluierungsfälle durchgeführt und dokumentiert?*
* *Wurde ein aussagekräftiges Beispiel für die Evaluierung herangezogen?*
* *Wurde der erstellte Prototyp zur Evaluierung herangezogen?*
* *Wurden alle signifikanten Daten der Evaluierung erfasst und rückverfolgbar gespeichert?*
* *Wurden alle wesentlichen Gesichtspunkte der Konzeption einer Evaluierung unterzogen?*
* *Ist das Gesamtergebnis der Evaluierung ausreichend für die Abnahme?*
* *Sind die Voraussetzungen der Prüfung, Prüfumgebung und Schritte der Prüfung aufgeführt?*

# Inhaltsverzeichnis

0 Inhaltsverzeichnis 3

1 Hyperparameter-Optimierung Modul B 4

1.1 Bezug zur Evaluierungsspezifikation 4

1.2 Evaluierungsprozedur 4

1.3 Ergebnis 5

1.4 Auswertung 7

2 Anzahl an Trainingsdaten pro Klasse 10

2.1 Bezug zur Evaluierungsspezifikation 10

2.2 Evaluierungsprozedur 10

2.3 Ergebnis 10

2.4 Auswertung 12

3 Finale Untersuchung Continual Learning 14

3.1 Bezug zur Evaluierungsspezifikation 14

3.2 Evaluierungsprozedur 14

3.3 Ergebnis 14

3.4 Auswertung 15

4 Literaturverzeichnis 17

# Hyperparameter-Optimierung Modul B

*Für jeden Prüffall ist ein Hauptgliederungspunkt mit den folgenden Unterpunkten anzu-legen. Es genügt die wichtigsten Fälle aufzuführen. Die Spezifikation muss nicht vollständig sein.*

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

## Bezug zur Evaluierungsspezifikation

*Benennung der entsprechenden Fälle in der Evaluierungsspezifikation. Genaue Bezeichnung der entsprechenden Systemkomponente(n).*

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 durchzuführen, was für die Gitter-Suche besonders relevant ist, da dort viele Tests für eine Abdeckung des Gitters erforderlich sind.

## Evaluierungsprozedur

*Kurze Beschreibung der Prozedur. Was sind die Voraussetzungen für die Prüfung? Wie sieht die Prüfumgebung aus? In welchen Schritten wird die Prüfung durchgeführt?*

Für die Hyperparameter-Optimierung wird zunächst eine zwei-dimensionale Gitter-Suche mit den Parametern und durchgeführt. Dafür werden diese beiden Parameter jeweils im Bereich in -er Schritten erhöht. Es werden jeweils 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 zusätzlich, dass die Gruppen von 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 für die Bestimmung der besten Parameter in jeder Wiederholung zufällig die benötigte Anzahl an Trainingsbilder 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 genutzt.

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 über die verschiedenen Fälle erstellt werden.

## Ergebnis

*Gegenüberstellung von erwartetem und erreichtem Ergebnis.*

Die Ergebnisse werden simultan für Split-MNIST und ImageNet-10 bewertet. Zunächst folgt dazu in einem ersten Schritt die Auswertung für die Gitter-Suche für die unterschiedlichen Werten von und . 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 für die Generalisierung und bei wird lediglich das neue Trainingssample genutzt und das alte 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 kaum eine Änderung mehr zu erkennen ist, da die Ähnlichkeit zwischen zwei Samples einer Klasse selten nahe 1 liegt. Dieser Schwellwert für , ab welchem es kaum Unterschiede in der Performanz gibt, wird für die unterschiedlichen Datensätze unterschiedlich erwartet. Ab diesem Schwellwert kann gesagt werden, dass jedes Sample, das im Training gesehen wird, als Repräsentation angelegt wird. Dies erfordert einen hohen Speicherbedarf und zudem sind die erhaltenen Repräsentationen nicht generalisiert, wodurch auf späteren abweichenden Test-Daten keine guten Generalisierungsfähigkeiten erwartet werden. Deshalb muss ein sinnvoller 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 dass positiv sein (wenn alle Samples der Klasse sehr ähnlich aussehen), aber im generellen ist das nicht wünschenswert, da dadurch abweichende Samples der Klasse (zum Beispiel eine verdrehte Zahl) nicht gut erkannt werden können. Somit muss für ein guter Mittelwert zwischen zu vielen und zu wenigen Repräsentationen gewählt werden.

In Abbildung 1 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.

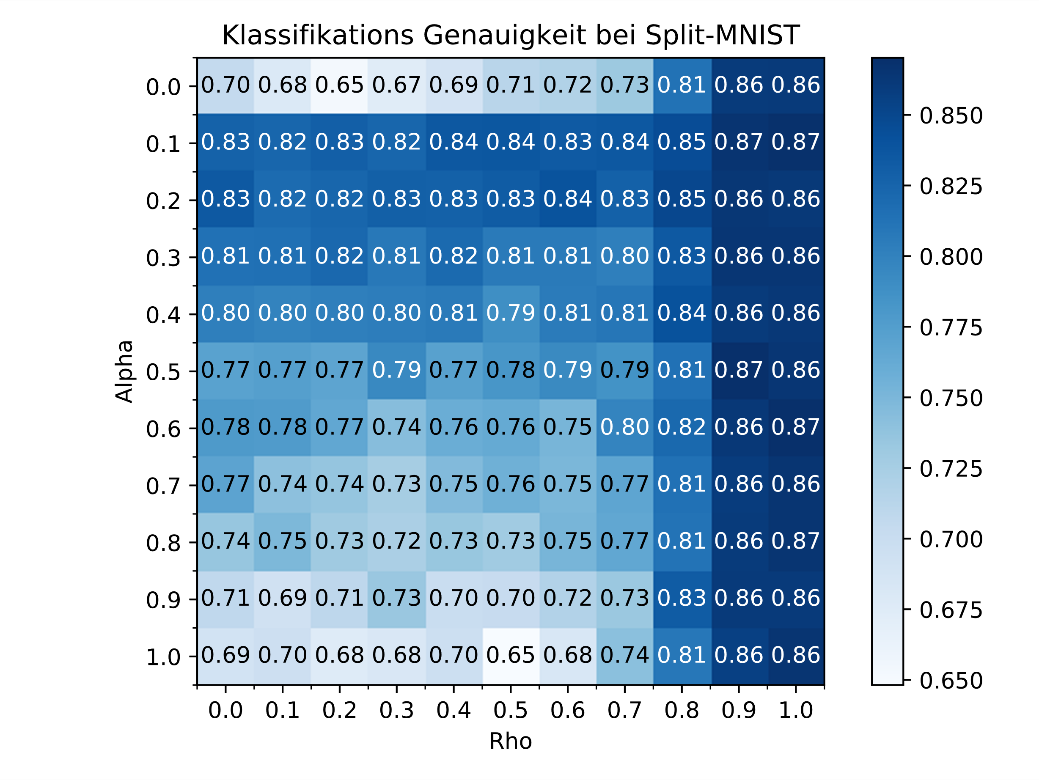


Abbildung : Ergebnisse der Gitter-Suche für und auf Basis des Split-MNIST Datensatzes

In Abbildung 2 sind die Ergebnisse für dieselben Testfälle auf Basis des ImageNet-10 Datensatzes dargestellt. Die Darstellungsart ist dabei identisch wie bereits für Abbildung 1 beschrieben.

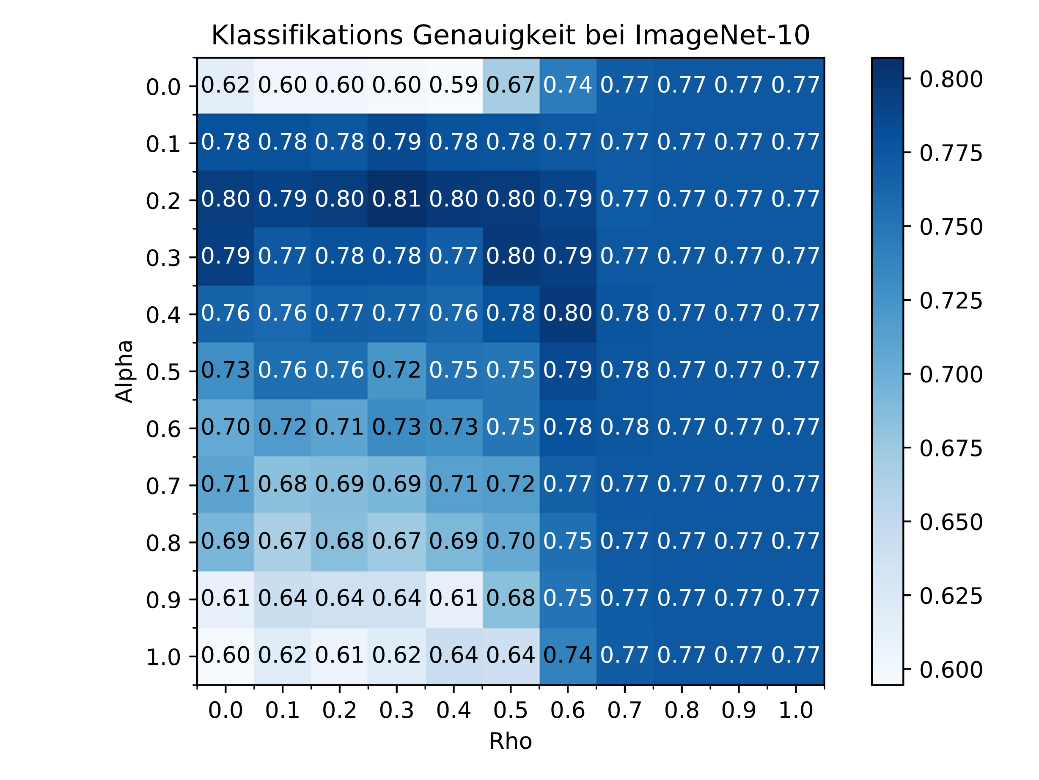


Abbildung : Ergebnisse der Gitter-Sucher für und auf Basis des ImageNet-10 Datensatzes

In Kapitel 1.4 werden die erzielten Ergebnisse nachfolgend ausgewertet und die passenden Hyperparameter für Modul B im Rahmen dieser Arbeit ausgewählt.

## Auswertung

*Mögliche Ursachen des Fehlverhaltens und vorgeschlagene Maßnahmen zur Be-seitigung*

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

Zunächst werden die Ergebnisse anhand des Parameters 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 lediglich das erste gesehene Sample genutzt wird, um die Repräsentation zu bilden. Anhand der in Abbildung 1 und Abbildung 2 dargestellten Ergebnisse lassen sich für die besten Ergebnisse erzielen. Es muss erwähnt werden, dass auch mit gut Ergebnisse erzielt wurden. 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 wird darüber auch bestimmt, 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. zu sehen, während bei ImageNet-10 dieser bereits bei zu beobachten ist. Dieser Unterschied kann mit der Komplexität der Bilder erklärt werden. Aufgrund der komplexen Bilder von ImageNet-10 sind die Ähnlichkeiten der Bilder geringer, wodurch bereits bei niedrigeren Werten von nahezu jedes Trainingssample als Repräsentation angelegt wird. Bei Split-MNIST sind die besten Ergebnisse mit den Werten und zu beobachten, also in den Fällen wo jedes Trainingssample als Repräsentation abgelegt wird. Bei ImageNet-10 sind die besten Ergebnisse für in dem Wertebereich zu sehen.

Bei Split-MNIST ist in Abbildung 1 das erwartete Ergebnis für zu sehen. 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, jedoch bei ca. 77% Klassifikationsgenauigkeit. Ein Grund dafür könnte in den Bildern des ImageNet-Datensatzes liegen. Dort sind die Objekte 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 denen von Vögeln sein, wenn beide im Himmel abgebildet sind mit viel blauem Hintergrund. Dadurch können sich einzelne, nicht generalisierte Repräsentationen, wie es für der Fall ist, den Validationsbildern von anderen Klassen ähneln, da einzelne Bilder der Klassen eventuell einen ähnlichen Feature-Vektor besitzen.

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 Speicherbedarf von Modul B für Split-MNIST und ImageNet-10 ist in Abbildung 3 und Abbildung 4 graphisch dargestellt.

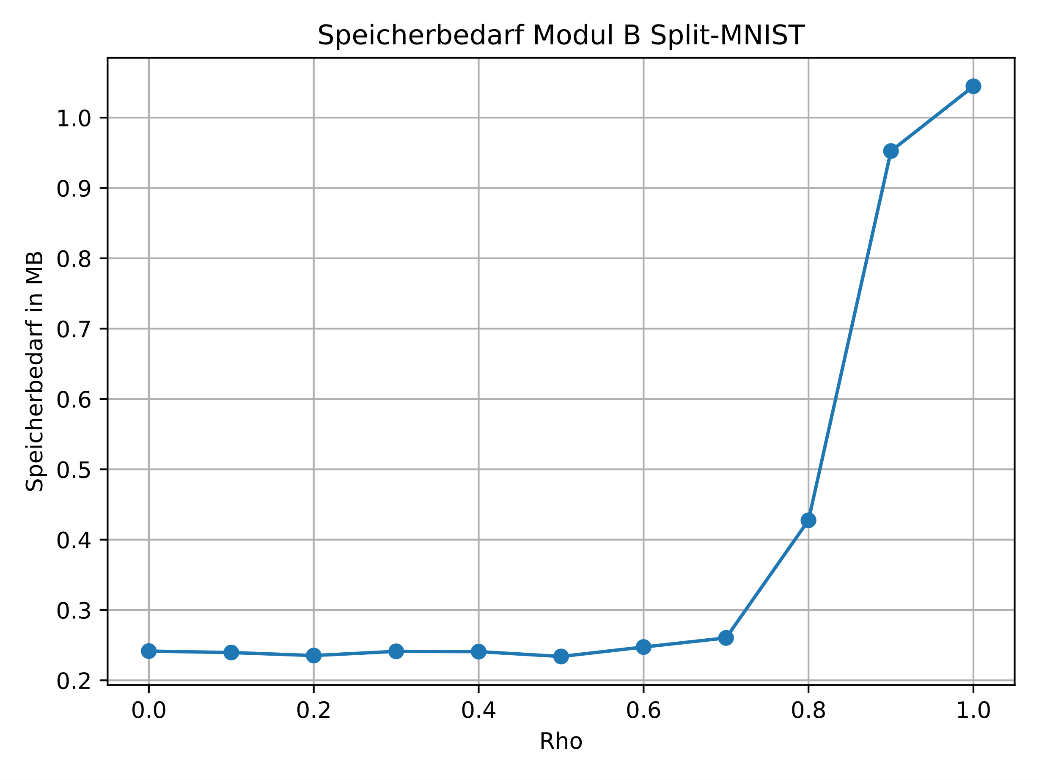


Abbildung : Speicherbedarf von Modul B in Abhängigkeit von für Split-MNIST

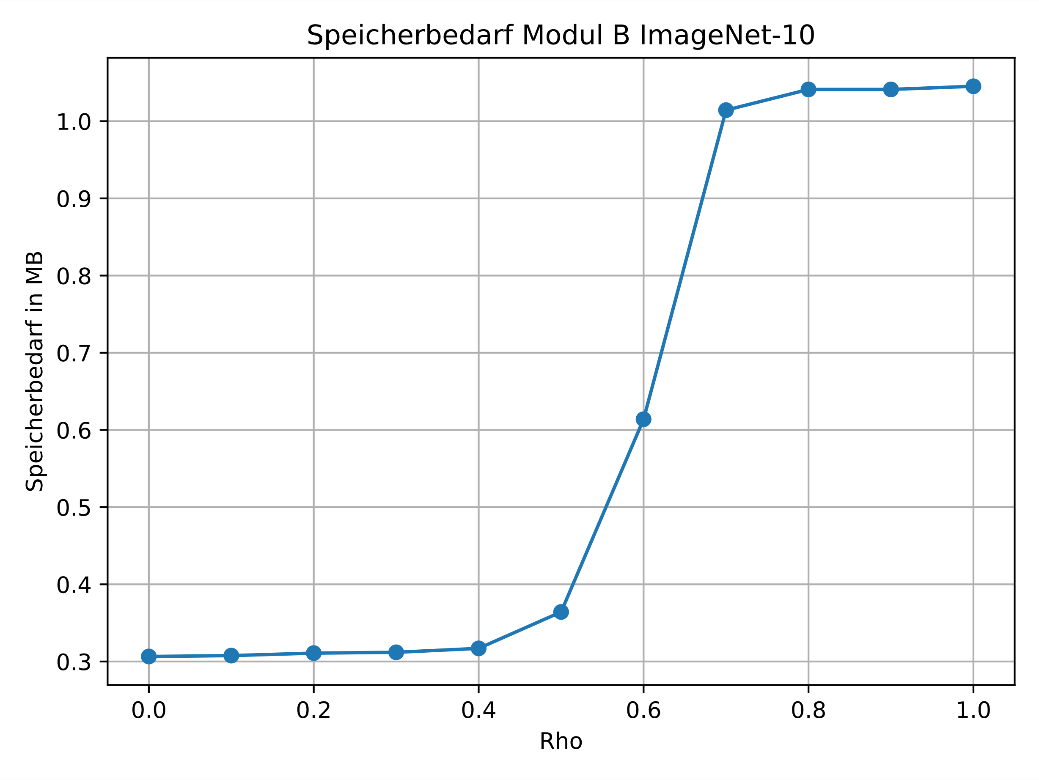


Abbildung : Speicherbedarf von Modul B in Abhängigkeit von für ImageNet-10

Bei diesen Abbildungen ist zu sehen, dass der Speicherbedarf ab dem Schwellwert (~0,9 für Split-MNIST und ~0,7 für ImageNet-10) deutlich zunimmt auf das 3- bist 4-fache des davor benötigten Speicherbedarfs. In diesem Beispiel ist der Speicherbedarf aufgrund der geringen Anzahl an Trainingsbildern (20 pro Klasse) immer noch gering mit ca. 1MB. Aber bei späteren Anwendungen mit mehr Klassen oder Trainingsbildern oder Trainingsdaten mit mehr Speicherbedarf kann dieser Faktor wichtig sein. Aufgrund dieser Ergebnisse wurde für der Wert 0,5 gewählt, da damit auf beiden Datensätzen eine gute Klassifikationsgenauigkeit erzielt werden kann und dennoch der Speicherbedarf gering ist mit ca. 250 KB für Split-MNIST und ca. 350 KB für ImageNet-10.

# Anzahl an Trainingsdaten pro Klasse

Schnell lernende inkrementelle Klassifikatoren sollen auf Basis von wenigen Beispieldaten einer Klasse diese erlernen können. Ein wichtiges Indiz ist dabei die Klassifikationsgenauigkeit über die gesehene Anzahl an Samples pro Klasse. Damit kann untersucht werden, ab wie vielen Trainingssamples eine Sättigung vorzufinden ist und wie viel Performanz erreicht werden kann, wenn mehr Trainingsbilder zur Verfügung stehen.

## Bezug zur Evaluierungsspezifikation

Wie in den Evaluierungskriterien der Evaluierungsspezifikation dargestellt, wird dieser Test genutzt, um eine Aussage über die Fähigkeit des Algorithmus zu erhalten, wie schnell neue Klassen erlernt werden können.

## Evaluierungsprozedur

Es werden die zuvor ermittelten Hyperparameter von Modul B verwendet. Mit diesen Parametern wird die als weitere Untersuchung die Anzahl an Trainingsbildern variiert. Die Anzahl der Trainingsbilder wird dabei auf Folgende Werte gesetzt: . Es werden pro Anzahl an Trainingsbildern 5 Wiederholungen durchgeführt, um einen aussagekräftigen statistischen Mittelwert und die Standardabweichung bilden zu können. Die weiteren einstellbaren Parameter sowie der Ablauf bleiben wie in Kapitel 1.2 beschrieben, und werden hier nicht nochmals dargestellt.

## Ergebnis

Es wird generell 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. Zudem wird erwartet, dass mit steigender Anzahl an Trainingsbildern pro Klasse die Varianz der Klassifikationsgenauigkeit abnimmt, und damit stabilere Ergebnisse erzielt werden können. Die Annahme ist, dass bei wenigen Trainingsbildern die spätere Klassifikationsgenauigkeit stark abhängig ist von der Auswahl dieser Trainingsbilder. Bei steigender Anzahl an Trainingsbilder sinkt die Abhängigkeit von den einzelnen Trainingsbilder. Zudem wird erwartet, dass der Speicherbedarf mit der Anzahl an Trainingsbildern steigt, da mehr Repräsentationen angelegt werden durch die höhere Zahl an gesehenen Trainingsdaten.

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 5 ist die Klassifikationsgenauigkeit für Split-MNIST dargestellt.

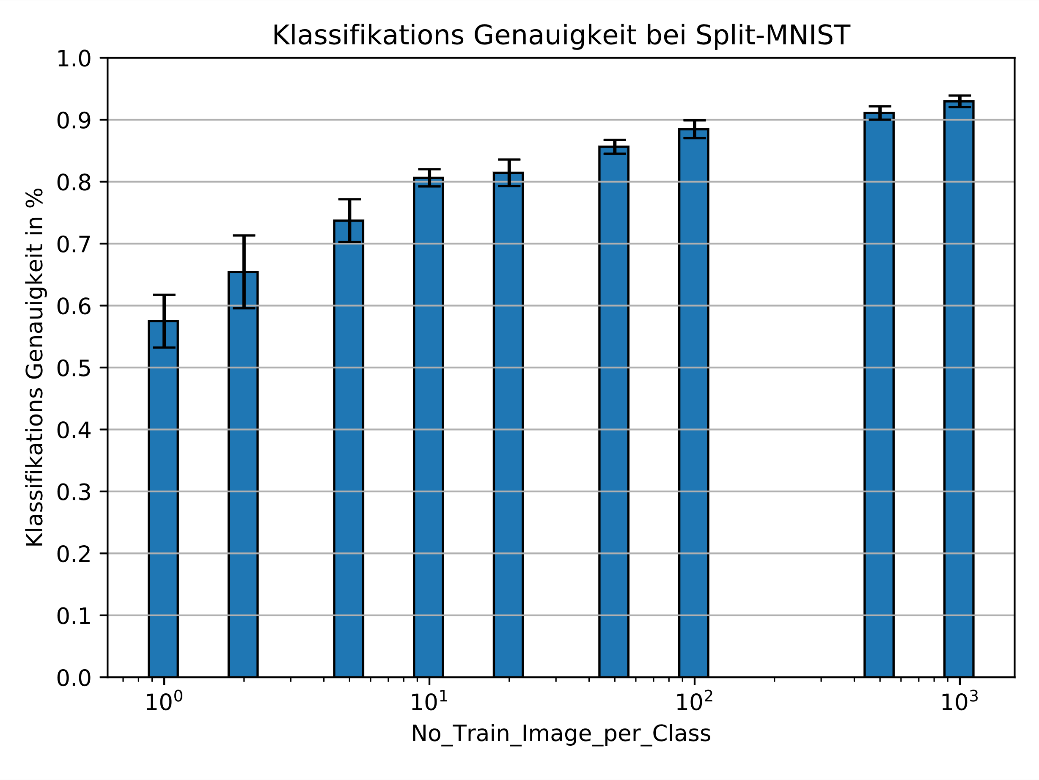


Abbildung : Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern Split-MNIST

Abbildung 6 stellt die Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern für ImageNet-10 dar.

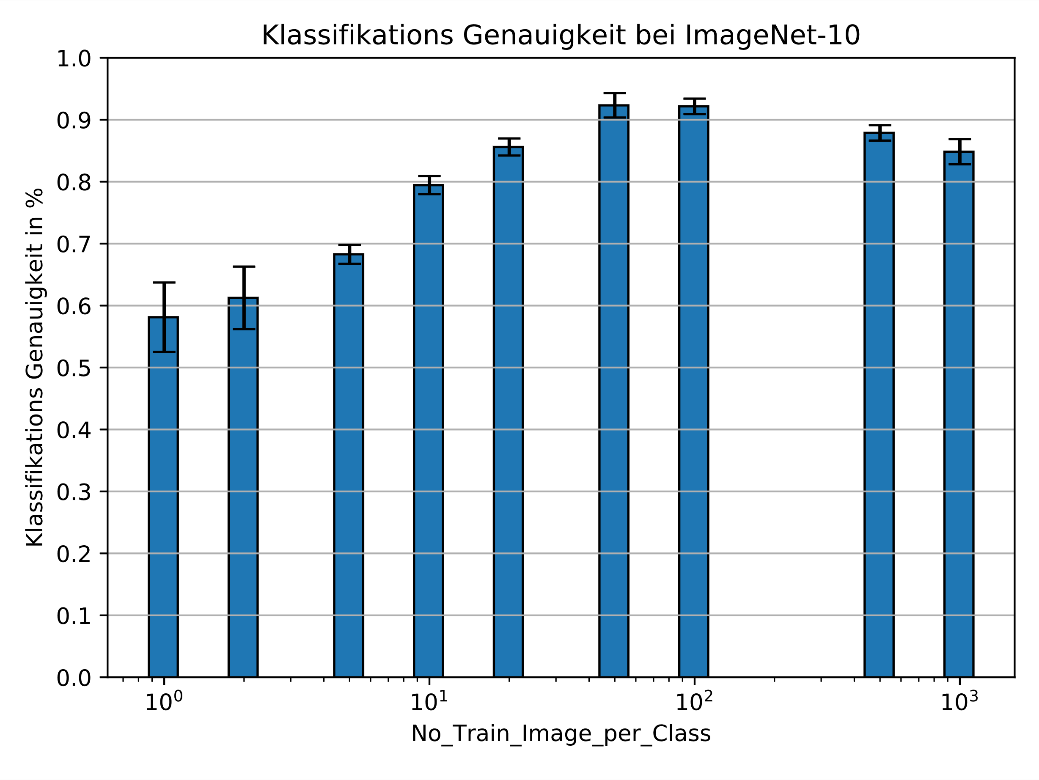


Abbildung : Klassifikationsgenauigkeit über die Anzahl an Trainingsbildern ImageNet-10

Dazugehörig wird der Speicherbedarf von Modul B für die jeweiligen Datensätze in Abhängigkeit von der Anzahl an Trainingsbildern in Abbildung 7 für Split-MNIST und in Abbildung 8 für ImageNet-10 dargestellt.

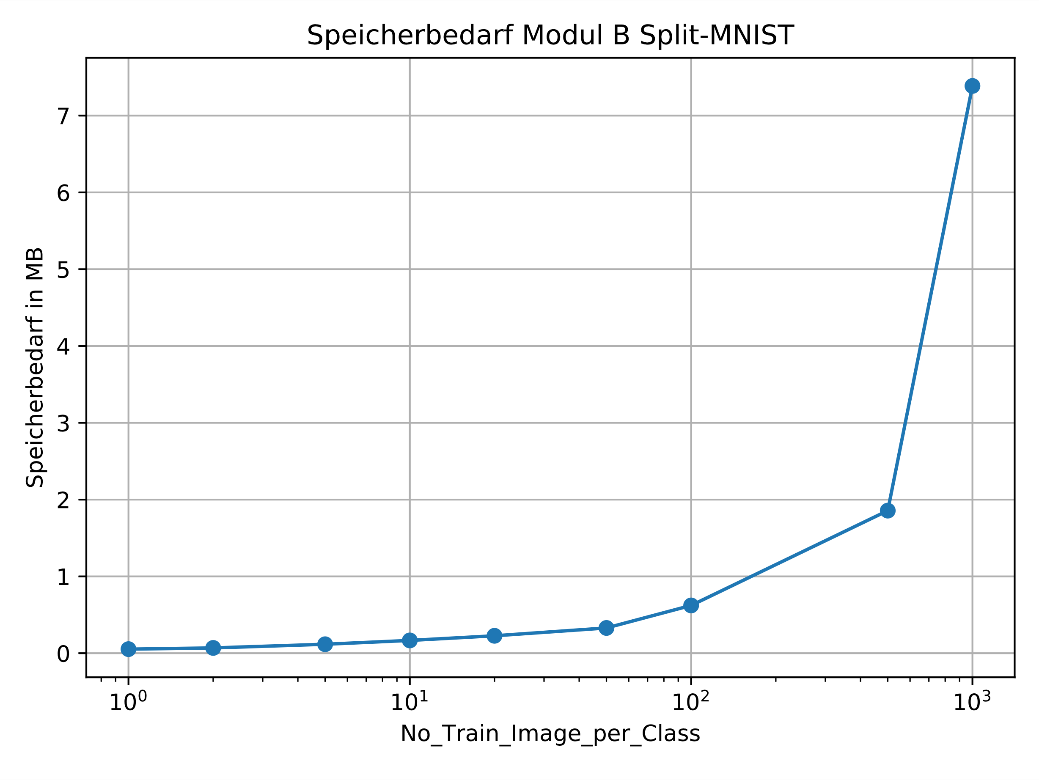


Abbildung : Speicherbedarf Modul B über die Anzahl an Trainingsbildern Split-MNIST

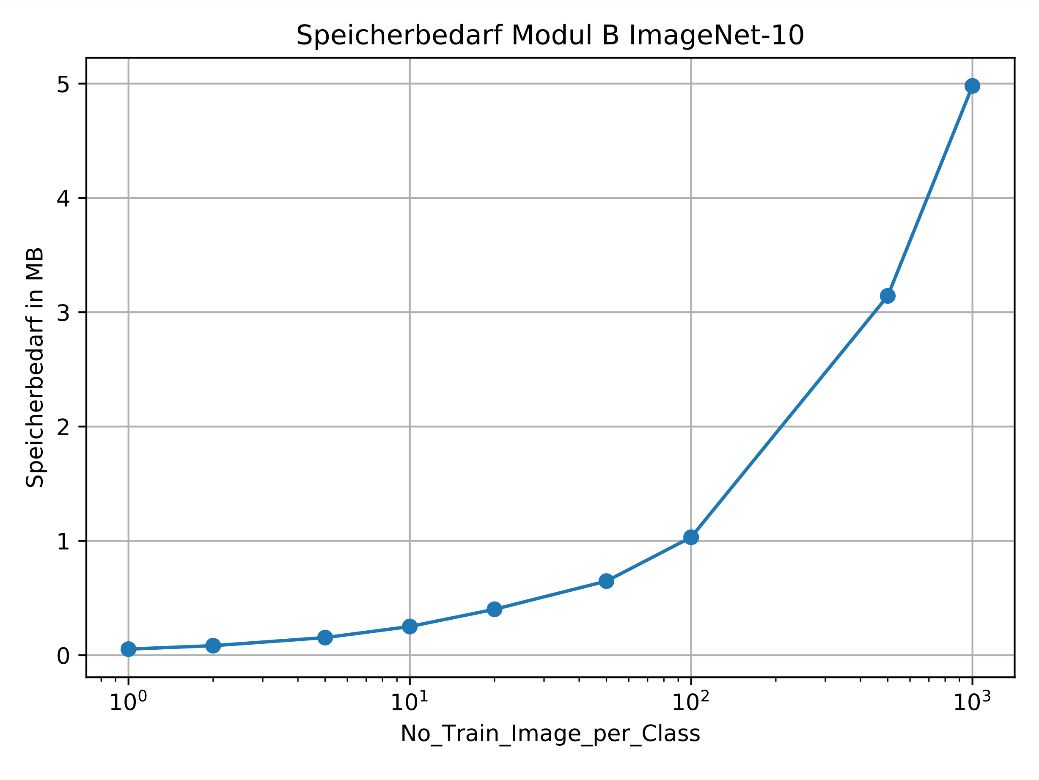


Abbildung : Speicherbedarf Modul B über die Anzahl an Trainingsbildern ImageNet-10

Die Bewertung der gezeigten Ergebnisse findet im folgenden Abschnitt statt.

## Auswertung

Zunächst kann die Annahme bestätigt werden, dass mit einer geringen Anzahl an Trainingsbildern eine hohe Varianz auftritt. Für Split-MNIST (Abbildung 5) beträgt die Varianz bei einem beziehungsweise zwei Trainingsbildern pro Klasse ca. 4,3 und 5,9 Prozentpunkte. Mit 500 und 1000 Trainingsbildern pro Klasse beträgt die errechnete Standardabweichung für 5 Wiederholungen 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 beziehungsweise zwei Trainingsbildern pro Klasse. Die geringste Abweichung ist für ImageNet-10 mit 100 und 500 Trainingsbildern pro Klasse zu erreichen, mit jeweils 1,2 Prozentpunkten.

Wenn nun 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 über fünf Wiederholungen eine Klassifikationsgenauigkeit von 92,96% bei einer Standardabweichung von +/-0,94 erreicht. Mit 500 Trainingsbildern pro Klasse kann ebenfalls bereits eine Klassifikationsgenauigkeit von 91,08% +/- 1,1 erreicht werden.

Wenn zusätzlich der Speicherbedarf in Abbildung 7 in Betracht gezogen wird, werden im weiteren Verlauf der Arbeit 500 Trainingsbilder pro Klasse für Split-MNIST genutzt. Denn der Speicherbedarf steigt in diesem Bereich 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 mit der leicht geringeren Genauigkeit gearbeitet werden, wenn dabei weniger als ein Drittel an Speicher benötigt wird.

Für ImageNet-10 lässt sich ein anderes Verhalten der Klassifikationsgenauigkeit über der Anzahl an Trainingsbildern 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 beschrieben Fall von zu vielen Repräsentation sein (siehe die Auswertung des Parameters in Kapitel 1.4 für Details). 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 (90,96% gegen 90,41%) werden 100 Trainingsbilder pro Klasse für weitere Untersuchungen mit dem ImageNet-10 Datensatz genutzt. Auch der Speicherbedarf ist in diesem Bereich noch akzeptabel, mit ca. 1 MB Speicherbedarf von Modul B für diese Anzahl an Trainingsbildern (Abbildung 8). Mit 50 Trainingsbildern pro Klasse wäre der Speicherbedarf ca. 0,65 MB und somit nicht entscheidend geringer.

Somit hat die Anzahl an Trainingsbildern pro Klasse wie erwartet einen direkten Einfluss auf die spätere Klassifikationsgenauigkeit. Bereits mit wenigen Trainingsbildern pro Klasse (z.B. 10) können akzeptable Klassifikationsgenauigkeiten von ca. 80% auf den Split-MNIST und ImageNet-10 Validationsdaten erreicht werden. Mit steigender Anzahl an Trainingsbilder nimmt die Klassifikationsgenauigkeit ebenso wie der Speicherbedarf des inkrementellen Klassifikator weiter zu. Je nach Anwendungsfall kann eine geringere Genauigkeit aufgrund weniger Trainingsdaten akzeptiert werden, wenn z.B. die Erzeugung von Trainingsdaten sehr kosten- und zeitaufwändig ist. Mit diesem Testfall kann gezeigt werden, dass in diesem Fall der hier untersuchte Algorithmus genutzt werden kann, da er bereits mit wenigen Trainingsbildern pro Klasse ansprechende Ergebnisse erzielt.

# Finale Untersuchung Continual Learning

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 jeweiligen Datensätze ermittelt, nachdem zuvor für die Untersuchungen der Hyperparameter eine Kreuzvalidation mithilfe des Trainings- und Validationsdatensatzes durchgeführt wurde.

## Bezug zur Evaluierungsspezifikation

Die in Kapitel 4 der Evaluierungsspezifikation genannten Testfälle werden nun durchgeführt. Es werden nun die Testdaten genutzt, nachdem zuvor für die Hyperparameteroptimierung Validationsdaten genutzt wurden, um eine korrekte Evaluierung der Ergebnisse zu gewährleisten. Somit werden nun Daten für die Erstellung der folgenden Metriken genutzt, die das Netzwerk bisher noch nicht gesehen hat. Dadurch kann eine Optimierung der Parameter auf Testdaten verhindert werden.

## Evaluierungsprozedur

Es werden die Testfälle des kontinuierlichen Lernens auf einem Gerät durchgeführt. Die Parameter werden auf Basis der vorherigen Testfälle ausgewählt und werden 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.

Zusätzlich zu den bisher untersuchten Datensätzen wird für die finale Bewertung dieser Test auch auf dem gesamten ImageNet-Datensatz mit 1000 Klassen durchgeführt. Dafür werden die Parameter auf dieselben Werte wie für ImageNet-10 gestellt.

## Ergebnis

In diesem Abschnitt werden die Ergebnisse dargestellt. Dafür wird für Split-MNIST eine Tabelle angelegt mit Ergebnissen anderer *Continual Learning*-Verfahren aus der Literatur sowie aktueller Top-Ergebnisse mit traditionellen Deep Learning Ansätzen. Für ImageNet-10 sind keine weiteren Ergebnisse von *Continual Learning*-Verfahren bekannt. Dieser Datensatz diente lediglich der Überprüfung des Potenzials auf komplexeren Eingangsdaten. Der Vergleich zu anderen Algorithmus und damit auch die Einordnung findet auf Basis des gesamten ImageNet-Datensatzes statt (1000 Klassen).

In Tabelle 1 sind die Ergebnisse für den Split-MNIST Datensatz dargestellt. Die erste Zeile ist dabei der hier untersuchte Algorithmus. Bei den weiteren Algorithmen ist jeweils über das Literaturverzeichnis angegeben, von welcher Quelle die genannte Genauigkeit kommt. Die Ergebnisse gelten für den Fall des inkrementellen Klassen Lernens nach dem kontinuierlichen Erlernen aller Klassen.

Tabelle : Klassifikationsgenauigkeit verschiedener Algorithmen auf Split-MNIST

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

EWC und SI stellen dabei typische Methoden des kontinuierlichen Lernens dar. Diese Methoden speichern keine Trainingsdaten und nutzen damit keine gespeicherten Repräsentationen zum Training. Dies wird in der Literatur auch Replay oder Rehearsal genannt. DGR nutzt diese Methode, in dem es komprimierte Repräsentationen der Trainingsdaten abspeichert. Wenn neue Klassen hinzukommen, werden aus den komprimiert gespeicherten Trainingsbilder der alten Klasse erzeugt (Generative) und in die neuen Trainingsdaten eingebracht. Das inkrementell trainierte MLP kann als untere Grenze gesehen werden, da hier *Catastrophic Forgetting* durch einfaches Anwenden 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 erreicht. Vergleichbare Ergebnisse für diesen Anwendungsfall sind in der Literatur nicht zu finden.

## Auswertung

Für Split-MNIST können die Ergebnisse gut 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 erzielen für das inkrementelle Klassen Lernen. Algorithmen die mit generativen Methoden arbeiten (wie DGR) erreichen jedoch eine bessere Klassifikationsgenauigkeit für diesen Anwendungsfall. Allerdings besitzen sie auch eine stark erhöhte Komplexität während des Trainings. Denn bei diesen Modellen müssen zusätzlich zu dem inkrementellen Klassifikator zusätzlich ein generatives Modell trainiert werden, welches ausgewählte Trainingsdaten komprimiert und wiederherstellt aus komprimierten Darstellungen. Zudem müssen diese komprimierten Darstellungen abgespeichert werden. All diese Punkte ermöglichen generativen Modellen eine bessere Klassifikationsgenauigkeit, jedoch sind diese Modelle aus den genannten Gründen (aktuell) nicht für den Einsatz auf einem mobilen Endgerät geeignet, was in dieser Arbeit ein wichtiger Auswahlpunkt für den Algorithmus war. Dennoch kommt der L DNN Algorithmus bereits in die groben Genauigkeitsbereiche des DGR, und eine Genauigkeit von ca. 87% auf Split-MNIST ist ein sehr bemerkenswertes Resultat.

Die Ergebnisse für ImageNet-10 können nicht mit anderen Algorithmen verglichen werden, jedoch kann damit geprüft werden, ob der Algorithmus auch auf komplexeren Eingangsdaten (hier 64x64 RGB-Bilder mit komplexen Klassen) funktioniert, bevor ein großer 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 bearbeiten kann.

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | 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. |
| [2] | 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. |