AG Angewante Informatik
Campus Velbert/Heiligenhaus
Fachbereich Elektrotechnik & Informatik

Hochschule Bochum
Bochum University
of Applied Sciences

KONTINUIRLICHES UND INKREMENTELLES LERNEN

AM BEISPIEL VON LEARN++

NAWRES JRIDI
GIANLUCA PODANN
NIKLAS DIEDERICH

Eingereicht am 9. April 2021

Hausarbeit im Rahmen der Veranstaltung
MASCHINELLES LERNEN UND DATA MINING
von Prof. Dr. JÖRG FROCHTE

Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit selbständig verfasst und keinen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Die Regelungen der geltenden Prüfungsordnung zu Versäumnis, Rücktritt, Täuschung und Ordnungsverstoß habe ich zur Kenntnis genommen.

Diese Arbeit hat in gleicher oder ähnlicher Form keiner Prüfungsbehörde vorgelegen.

Dortmund, den 09.04.2021

Unterschrift

Wermelskirchen, den 09.04.2021

Unterschrift

Essen, den 09.04.2021

Unterschrift

Inhaltsverzeichnis

1 Einleitung

1.1 Allgemeines

In der folgenden Hausarbeit werden die Grundlagen des inkrementellen und kontinuierlichen Lernens erklärt. Diese sind die Basis für den Learn++ Algorithmus, welcher das inkrementelle Lernen implementiert. Im weiteren Verlauf wird der Learn++ Algorithmus genauer vorgestellt und in Programmcode umgesetzt. Der Code wird anschließend mit dem CIFAR10 Datensatz, welcher aus 60000 kleinen Bildern besteht getestet und das Ergebnis mit der Klassifizierung eines klassischen neuralen Netzes auf demselben Datensatz verglichen.
Die Grundlagen dafür sind die aus der Vorlesung bekannten convolutional neural networks, kurz CNN und der von uns implementierte Algorithmus Learn++.Dieser basiert auf den Grundlagen des kontinuierlichen und des inkrementellen Lernens, weiterhin wird ein Vergleich zwischen maschinellen Lernen und menschlichen Lernen gezogen.Dabei geht es im Grundsatz darum, dass maschinelles Lernen das sogenannte katastrophale Vergessen beinhaltet, also dass von einem zum nächsten Lernschritt, das vorherige Wissen überschrieben wird, im Gegensatz dazu lernt der Mensch, indem er sein Wissen erweitert. Der Learn++ Algorithmus ist nach dem inkrementellen Lernen aufgebaut, welches das katastrophale Vergessen verhindern soll, außerdem ist er darauf ausgelegt neue Klassen gut klassifizieren zu können.

1.2 Aufgabenverteilung

Kapitel	**Autor**
1 Einleitung	Frau Jridi &Herr Podann & Herr Diederich
2.1 Inkrementelles und kontinuierliches Lernen	Frau Jridi
2.2 Learn++	Herr Podann
3.1 Datensatz	Herr Podann
3.2 Umsetzung des Learn++ Algorithmus	Herr Diederich
3.3 Neuronales Netz	Herr Podann
4.1 Ergebnisse Learn++	Herr Diederich
4.2 Ergebnisse CNN	Herr Podann
5.1 Fazit	Herr Diederich
5.2 Ausblick	Herr Diederich

Tabelle 1.1: Verteilung der Aufgaben

2 Grundlagen

2.1 Inkrementelles und kontinuierliches Lernen

2.1.1 kontinuierliches Lernen

kontinuierliches Lernen oder, wie es auch genannt wird, Lifelong learnen bedeutet für uns Menschen, wörtlichvon der Wiege bis zur Bahre, also es soll in allen Phasen des Lebenszykluses stattfinden. In neueren Versionen heißt es auch, dass es lebensumfassend sein sollte, das heißt, es ist eingebettet in allen Lebenskontexten von der Schule bis zum Arbeitsplatz, Zuhause/ von Zuhause aus und in der Gemeinschaft.
LifeLong Lernen ist der kontinuierliche Aufbau von Fähigkeiten und Wissen im Laufe des Lebens, der durch Erfahrungen im Lebensverlauf erworben wird.
Außerdem geht es beim LifeLong Lernen darum, zweite Chancen zu bieten, um grundlegende Fähigkeiten aufzufrischen und auch Lernmöglichkeiten auf fortgeschrittenerem Niveau anzubieten. All dies bedeutet, dass die formalen Systeme der Bereitstellung viel offener und flexibler werden müssen, damit solche Möglichkeiten wirklich auf die Bedürfnisse des Lernenden bzw. des potenziellen Lernenden zugeschnitten werden können.
Im Kontext des maschinellen Lernens bedeutet dies, dass das Vorhersagemodell reibungslos aktualisiert werden kann, um unterschiedliche Aufgaben und Datenverteilungen zu berücksichtigen, aber dennoch nützliches Wissen und Fähigkeiten im Laufe der Zeit Wiederzuverwenden und beibehalten zu können.

2.1.2 Inkrementelles Lernen

Inkrementelles Lernen bezieht sich auf ein lernendes System, das kontinuierlich neues Wissen aus neuen Mustern lernen kann und den Großteil, des zuvor gelernten Wissens, beibehalten kann. Inkrementelles Lernen ist ein spezielles Szenario der maschinellen Lerntechnologie, das sich mit Anwendungen befassen kann, die dem menschlichen Verhalten und Denken eher entsprechen.
Das klassische maschinelle Lernmodell wird mit statischen, identisch verteilten und gut beschrifteten Trainingsdaten gelernt. Die äußere Umgebung der realen Welt ändert sich jedoch dynamisch, so dass der intelligente Agent die Fähigkeit haben muss, kontinuierlich zu lernen und sich das neue Wissen zu merken. Ein inkrementelles Lernmodell oder ein Modell des lebenslangen Lernens kann neues Wissen erlernen und das alte in lebenslang beibehalten. Inkrementelles Lernen erfolgt in Szenarien. zum besseren Verständnis werden inkrementelle Lernszenarien in aufgabeninkrementelle Szenarien, domäneninkrementelle Szenarien und klasseninkrementelle Szenarien nach bestimmten Versuchsprotokollen unterteilt. Abbildung 2.1 stellt den Prozess vom inkrementellen Lernen dar. Die kontinuierlich erfassten Daten werden in sequenzielle Tasks unterteilt, die durch T1, T2, . . .TN dargestellt werden können, so dass

$$Ti = (xi1, yi1), (xi2, yi2), ...$$

Ti ist der i-te Datensatz, Mi ist das inkrementelle Lernmodell, es wird mit den neuen Daten Ti und dem Modell Mi-1 trainiert, wie in der Gleichung gezeigt.

$$Mi = f(Ti, Mi - 1)$$

Beim Lernen mit neuem Wissen kommt es zu einem deutlichen Abfall der Modellleistung bei bereits gelerntem Wissen. Dies wird als katastrophales Vergessen bezeichnet, das die langjährige Herausforderung des inkrementellen und lebenslangen Lernens darstellt. [YLM20]

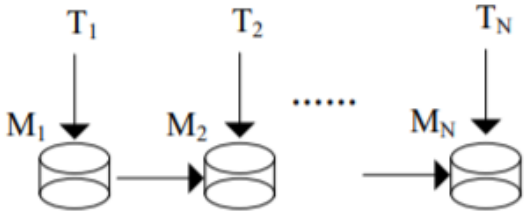

Abbildung 2.1: Prozess des inkrementellen Lernens

2.1.3 Katastrophales Vergessen

McCloskey und Cohen sind zwei Forscher, die 1989 entdeckten, dass ein neuronales Netzwerk beim Training neuer Aufgaben, die in zuvor trainierten Aufgaben gelernten Fakten vernachlässigt, was als katastrophales Vergessen bezeichnet wird.[Fre99]
In neuronalen Netzen bedeutet dies, dass eine neuartige Aufgabe wahrscheinlich die in der Vergangenheit gelernten Gewichte auslöscht und damit die Leistung des Modells für vergangene Aufgaben verringert. Dieses Problem ist sehr ernst, und wenn es nicht gelöst wird, kann sich ein isoliertes neuronales Netzwerk nicht an ein Szenario des lebenslangen Lernens anpassen, da es beim Lernen neuer Dinge das vorhandene Wissen vergisst. Dies wird als das Stabilitäts-Plastizitäts-Dilemma bezeichnet.[GIP19]
Das katastrophale Vergessen ist in beiden Fällen ein Problem, Einerseits kann das Modell zu stabil sein, so dass es nicht in der Lage ist, innovative Fakten aus zukünftigen Trainingsdaten auszunutzen. andererseits kann das Modell mit ausreichender Plastizität, auch unter großen Gewichtsänderungen leiden und im Voraus gelernte Repräsentationen vergessen.

2.1.4 Biologische Aspekte

Kontinuierliches Lernen und Inkrementelles Lernen funktionieren wie das Gehirnsystem eines Organismus und ist eines der Endziele von Systemen der künstlichen Intelligenz.

Abbildung 2.2: Menschen Gehirn

Neurowissenschaftler haben eindeutig gezeigt, dass das Gehirn von Menschen eine große Fähigkeit besitzt, sich an die Anforderungen seiner Umwelt anzupassen: die Plastizität. Neuronale Verbindungen werden neu gebildet oder verstärkt, andere werden geschwächt oder beseitigt, je nach Bedarf. Das Ausmaß der Veränderung hängt von der Art des Lernens ab. Langfristiges Lernen führt zu tiefgreifenderen Veränderungen. Es hängt auch davon ab, wann das Lernen stattfindet. Bei Säuglingen werden neue Synapsen mit einer außerordentlichen Geschwindigkeit gebildet. [Ull]

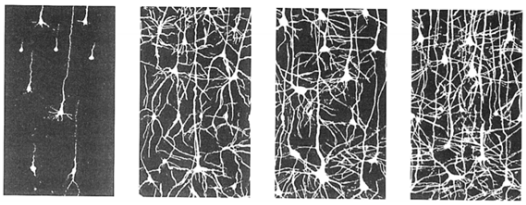

Abbildung 2.3: Vernetzungen beim Menschen nach der Geburt, nach 3 Monaten, nach 15 Monaten

Der Mensch zeigt eine überlegene (außergewöhnliche) Fähigkeit, in kontinuierlichen Umgebungen zu lernen. Dank der langen Fortschritte in der neurophysiologischen Entwicklung hat das Gehirn die Fähigkeit, Wissen inkrementell in aufeinanderfolgenden Aufgaben zu erwerben und zu speichern. [Sch]
Daher inspirieren die biologisch untersuchten Prinzipien der Wissensverarbeitung im Gehirn die Entwicklung von computergestützten Ansätzen.
Es gibt Mechanismen, die das Gleichgewicht zwischen Stabilität und Plastizität von Hirnarealen regulieren und kognitive Systeme entwickeln sich als Reaktion auf externe Stimulation.Deshalb muss ein Modell sowohl plastisch sein, um das KATASTROPHALE VERGESSEN zu überwinden und um neues Wissen zu erwerben, als auch stabil, um vorhandenes Wissen zu konsolidieren. Diese beiden Anforderungen gleichzeitig zu erfüllen ist sehr schwierig. Dieses Phänomen wird als das Stabilität-Plastizität-Dilemma bezeichnet. Dies ist eine erstaunliche Fähigkeit zur Anpassung. Sie kann Wissen und Fähigkeiten effektiv erwerben, und basiert darauf neue Erfahrungen zu verbessern und diese in mehrere Bereiche zu übertragen. [GIP19]
Die Hebbsche Plastizität hat auch Forscher inspiriert, um zu beschreiben, wie Neuronen auf externe Reize reagieren. Es wird davon ausgegangen, dass wenn ein Neuron die Aktivität eines anderen stimuliert, die Verbindung zwischen ihnen verstärkt wird.
Hebbsche Plastizität kann durch synaptische Beschränkungen und Rückkopplungssignale verstärkt werden. Die Theorie des komplementären Lernsystems zeigt, dass der Hippocampus durch kurzfristige Anpassungsfähigkeit schnell lernt, während die Großhirnrinde durch das Langzeitgedächtnis nur langsam lernen kann [Kar]
Die einfachste Form der Hebbschen Plastizität betrachtet eine synaptische Stärke w , die durch das Produkt aus präsynaptischer Aktivität x und postsynaptischer Aktivität y aktualisiert wird.
$\delta w = m - x - y - n.$
n ist eine vorgegebene Lernrate und m ein zusätzliches Modulationssignal. [GIP19]
Diese Hebb-Regel besagt, dass eine wiederholte und anhaltende Stimulation der postsynaptischen Zelle durch die präsynaptische Zelle zu einer Steigerung der synaptischen Wirksamkeit führt. Die präsynaptische Zelle führt zu einer erhöhten synaptischen Wirksamkeit.
Während des Entwicklungsprozesses stabilisieren sich neuronale Systeme, um optimale funktionelle Muster der neuronalen Konnektivität zu bilden.

2.1.5 Methode Beschreibung

Das katastrophale Vergessen ist die bedeutsame Herausforderung für die neuartigen Lernmethoden. Viele Wissenschaftler haben im Laufe der Jahre durch eingehende Forschung versucht, diese katastrophale Vergessen bei Maschinen zu verringern. Daher werden drei Arten von Lösungsmethoden vorgeschlagen. Zunächst gibt es die Regulierungsmethoden, welche das gesamte Netzwerk neu trainieren und dabei regulieren, um katastrophales Vergessen mit zuvor gelernten Aufgaben zu verhindern. Als nächstes die architektonischen Methoden, die das Netzwerk selektiv trainieren und bei Bedarf erweitern, um neue Aufgaben zu repräsentieren. Und zum Schluss Methoden, die komplementäre Lernsysteme zur Gedächtniskonsolidierung modelliere, z.B. durch die Verwendung von Memory Replay zur Konsolidierung interner Repräsentationen dienen.

Regularisierungsmethoden

Diese Methode mildert das katastrophale Vergessen durch Hinzufügen eines speziellen Regularisierungsterms zur Verlustfunktion ab. Die zentrale Idee besteht darin, die Aktualisierung der neuronalen Gewichte zu begrenzen, um die Stabilität des Modells zu verbessern und damit das katastrophale Vergessen zu vermindern.
Diese Methode ist im Allgemeinen von theoretischen Modellen aus den Neurowissenschaften inspiriert, die vorschlagen, dass konsolidiertes Wissen vor dem Vergessen geschützt werden kann, indem Synapsen eine Kaskade von Zuständen aufweisen, die verschiedene Stufen der Plastizität erzeugen . Aus rechnerischer Sicht wird dies typischerweise durch zusätzliche Regularisierungsterme modelliert, die Änderungen in der Abbildungsfunktion eines neuronalen Netzes bestrafen.[GIP19]

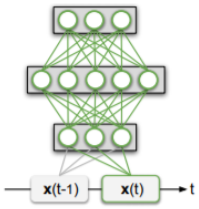

Abbildung 2.4: retraining mit Regularisierung

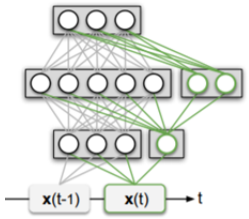

Abbildung 2.5: Training mit Netzwerk Expansion

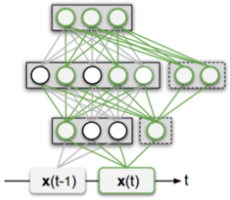

Abbildung 2.6: Selektives retraining und expansion des Netzwerks

Einige Forscher schlugen den Ansatz des Lernens ohne Vergessen vor, der aus Convolutional neuronalen Netzen (CNN) besteht, bei dem das Netz mit Vorhersagen der zuvor gelernten Aufgaben erzwungen wird, dem Netz mit der aktuellen Aufgabe ähnlich zu sein, indem Wissens destillation verwendet wird, d. h. die Übertragung von Wissen aus einem großen, stark regulierten Modell in ein kleineres Modell [YLM20]. Gemäß dem Lernen ohne Vergessen-Algorithmus werden bei einem Satz gemeinsamer Parameter s über alle Aufgaben hinweg die Parameter der neuen Aufgabe n zusammen mit s optimiert, wobei die zusätzliche Einschränkung gilt, dass sich die Vorhersagen auf den Stichproben der neuen Aufgabe unter Verwendung von s und den Parametern der alten Aufgaben o nicht signifikant verschieben, um o zu behalten. Angesichts der Trainingsdaten für die neue Aufgabe (Xn, Yn), der Ausgabe alter Aufgaben für die neuen Daten Yo und zufällig initialisierter neuer Parameter n sind die aktualisierten Parameter $\boldsymbol{\theta_s}$, $\boldsymbol{\theta_o}$, $\boldsymbol{\theta_n}$, gegeben .[CL18]

Lernen ohne Vergessen

Input : θ_s Set von Parametern, die von allen Tasks gemeinsam genutzt werden, θ_o Set von Parametern, die speziell für vorherige Aufgaben gelernt wurden, Training data X_n , Y_n für die neue Task.

Output : updated Parametern θ_s^* , θ_o^* ,θ_n^*

1: // Initialization phase.
2: $Y_0 \leftarrow$ CNN (X_n, θ_s, θ_o)
3: $\theta_n \leftarrow$RANDINIT ($|\theta_n|$)
4: // Training phase.
5: Define $\hat{Y}_n \equiv$ CNN(X_n ,$\hat{\theta}_s$, $\hat{\theta}_n$)
6: Define $\hat{Y}_o \equiv$ CNN (X_n, $\hat{\theta}_s$, $\hat{\theta}_o$)
7: θ_s^* , θ_o^* ,θ_n^* $\leftarrow$ $argmin_{\hat{\theta}s,\hat{\theta}o,\hat{\theta}n}(\mathcal{L}_{new}(\hat{Y}_n, Y_n) + \lambda_o \mathcal{L}_{old}(\hat{Y}_o, Y_0) + \mathcal{R}(\theta_s, \theta_o, \theta_n)$

Abbildung 2.7: Algorithmus Lernen ohne Vergessen

Lnew $(\hat{Y}_n, Y_n)$, $Lold(\hat{Y}_o, Y_0)$ minimizieren die Differenz zwischen den vorhergesagten Werten $\hat{Y}$ und den"ground-truth" -Werten Y der neuen bzw. alten Aufgaben unter Verwendung $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$, $\hat{\boldsymbol{\theta}}_{\boldsymbol{o}}$, $\hat{\boldsymbol{\theta}}_{\boldsymbol{n}}$, λ_o wird verwendet zum Ausgleich neue/alte Aufgaben, und R ist ein Regularisierungsterm, um eine Überanpassung zu verhindern. Dieser Ansatz hat jedoch den Nachteil, dass er stark von der Relevanz der Aufgaben abhängt und dass die Trainingszeit für eine Aufgabe linear mit der Anzahl der gelernten Aufgaben ansteigt.
Eine andere Forschung schlug vor, den l2-Abstand zwischen den endgültigen versteckten Aktivierungen zu regularisieren und dabei die zuvor gelernten Input-Output-Mappings zu bewahren, indem zusätzliche Aktivierungen mit den Parametern der alten Aufgaben berechnet werden. Diese Ansätze sind jedoch rechenintensiv, da sie die Berechnung der Parameter der alten Aufgaben für jede neue Datenprobe erfordern. Andere Ansätze entscheiden sich dafür, entweder die Aktualisierung von Gewichten, die auf alten Aufgaben trainiert wurden, komplett zu verhindern oder die Lernrate zu reduzieren, um signifikante Änderungen der Netzwerkparameter während des Trainings mit neuen Daten zu vermeiden. Es kann gesagt werden, dass die Regularisierungs Methode eine Möglichkeit bietet, das katastrophale Vergessen unter bestimmten Bedingungen abzumildern. Sie beinhaltet jedoch zusätzliche Verlustterme zum Schutz von konsolidiertem Wissen, was bei einer begrenzten Menge an neuronalen Ressourcen zu einem Kompromiss bei der Leistung von alten und neuen Aufgaben führen kann.

architektonische Methoden

Bei dieser Methode werden die architektonischen Eigenschaften als Reaktion auf neue Informationen geändert, indem neue neuronale Ressourcen dynamisch angepasst werden, z. B. durch erneutes Training mit einer erhöhten Anzahl von Neuronen oder Netzwerk schichten. Learn++, das wir in unserem Projekt anwenden werden, ist ein gutes Beispiel für diese Methode,Learn++ trainiert zunächst mehrere Klassifikatoren mit unterschiedlichen Training-subsets. Dann trifft es Entscheidungen unter Verwendung einer schwachen Klassifikator-Kombination, die auf einem adaptiven neuronalen Netzwerks.Als ein früher vorgeschlagener Algorithmus ist die Fähigkeit, neue Klassen zu lernen, eines der Hauptmerkmale und Vorteile von Learn++. Er hat auch die Vorteile einer kleinen Anzahl von Parametern und einer kurzen Trainingszeit. Außerdem, Ein Modell der progressiven Netzwerke wurde von einer Forscher vorgeschlagen. Die Parameter eines durch die vorangegangenen Aufgaben trainierten Netzes sind festgelegt, um ein katastrophales Vergessen zu vermeiden. Beim Training einer neuen Aufgabe bringt das Progressive Netzwerk die Erfahrung des zuvor gelernten Wissens ein, indem es die Ausgabe des vorherigen Netzes berücksichtigt. Der Aufbau des progressives Netzwerk ist in Abbildung 2.8 dargestellt. Das progressive Netzwerk behält die Struktur bei, die trainiert wurde, um die Leistung des Modells bei der alten Aufgabe zu schützen und das katastrophale Vergessen effektiv zu mindern. Die beiden Säulen auf der linken Seite (gestrichelter Pfeil) wurden jeweils für das Training von Aufgabe 1 und Aufgabe 2 verwendet. Das mit a markierte graue Kästchen stellt seitliche Verbindungen, um die Ausgabe des vorherigen Netzes zu erhalten. Die Spalte ganz rechts wird für die letzte Aufgabe hinzugefügt, die Zugriff auf alle zuvor gelernten Merkmale hat.
In Experimenten wurden gute Ergebnisse bei einer Vielzahl von Reinforcement-Learning-Aufgaben erzielt, die die üblichen Ansätze übertreffen, die entweder ein Pre-Training durchführen oder die Modelle inkrementell verfeinern, indem sie nur bei der Initialisierung Vorwissen einbeziehen. Intuitiv vermeidet dieser Ansatz katastrophales Vergessen, führt aber dazu, dass die Komplexität der Architektur mit der Anzahl der gelernten Aufgaben wächst.[AAR16]

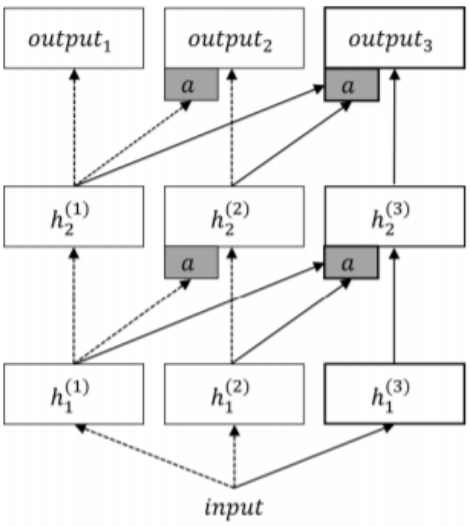

Abbildung 2.8: Die Struktur eines progressiven neuronalen Netzes

Andere Forschungen haben gezeigt, dass lebenslanges Lernen menschlicher Handlungs sequenzen im Sinne einer vorhersage gesteuerten neuronalen Dynamik erreicht werden kann, wobei interne Repräsentationen in einer Hierarchie rekurrenter selbstorganisierender Netzwerke entstehen. Selbstorganisierende Netzwerke können neuronale Ressourcen dynamisch zuweisen und Konnektivitäts muster nach dem kompetitiven

Hebbschen Lernen aktualisieren. Jedes Neuron in der neuronalen Karte besteht aus einem Gewichts vektor und einer Anzahl von Kontext deskriptoren mit. Daher werden die rekurrenten Neuronen in der Karte selektive Sequenz-Schnappschuss-Prototypen der Eingabe kodieren. Jedes Neuron ist mit einem Gewöhnungs zähler ausgestattet, der die Häufigkeit angibt, mit der es feuert, basierend auf einer Vereinfachung der Verringerung der Effizienz einer habituierenden Synapse über die Zeit. Das Netzwerk wird mit zwei Neuronen initialisiert und fügt bei jeder Lern iteration ein neues Neuron ein, wenn die Netzwer kaktivität eines habituierten Neurons unter einem vorgegebenen Schwellenwert liegt. Die Forscher zeigten, dass selbstorganisierende Netzwerke mit additiver Neurogenese besser abschneiden als ein statisches Netzwerk mit der gleichen Anzahl von Neuronen. Das bietet einen Einblick in das Design von neuronalen Architekturen in inkrementellen Lernszenarien, wenn die Gesamtzahl der Neuronen festgelegt ist.
Es gibt in der Tat viele andere Forschungen, die ähnliche Methoden vorgeschlagen haben, die auf einem dynamischen architektonischen Design basieren und helfen, dieses katastrophale Versehen zu reduzieren.

Komplementäre Lernsysteme und Gedächtniswiedergabe (CLS)

Die CLS-Theorie liefert die Grundlage für ein Rechenmodell der Gedächtnis integration und des Gedächtnis rückrufs. In diesem Modell werden die komplementären Aufgaben des Gedächtnisses und der Generalisierung durch die Interaktion zwischen dem Hippocampus von Säugetieren und dem Neokortex dargestellt. Diese Interaktion zwischen episodischem Gedächtnis (spezifische Erfahrung) und semantischem Gedächtnis (strukturiertes Allgemeinwissen) liefert wichtige Informationen über die Mechanismen der Wissenskonsolidierung in Abwesenheit von sensorischen Hinweisen. Zur Bekämpfung des katastrophalen Vergessens werden Duale Gedächtnis-Lernsysteme verwendet, die in unterschiedlichem Maße von der CLS-Theorie inspiriert sind. Ein erstes rechnerisches Beispiel für dieses Konzept wurde vorgeschlagen, bei dem jede synaptische Verbindung zwei Gewichte hat: ein plastisches Gewicht mit langsamer Änderungs rate, das langfristiges Wissen speichert, und ein Gewicht mit schneller Änderungs rate für temporäres Wissen.
Diese Methode der doppelten Gewichtung spiegelt die Eigenschaften von komplementären Lernsystemen wider, um katastrophales Vergessen während des sequentiellen Aufgaben lernens zu vermindern. Andere Forscher haben Rehearsal- und Pseudo-Rehearsal-Strategien verwendet, die einer relativ einfachen Idee, der Retrospektion, folgen, um mit katastrophalem Vergessen umzugehen. Robins stellte fest, dass das katastrophale Vergessen durch Proben oder Pseudo proben gemildert werden kann. Bei der Pseudo repetition werden die Trainings muster nicht explizit im Speicher gehalten, sondern aus einem probabilistischen Modell gezogen. Ein Grund für das katastrophale Vergessen ist, dass beim inkrementellen Lernen die entsprechende Kontrolle des Vorwissens fehlt. Wenn ein Modell beim Erlernen von neuem Wissen Vorwissen abrufen kann, kann es das katastrophale Vergessen abmildern.

2.2 Learn++

Der Learn++ Algorithmus basiert auf inkrementellem Lernen, entsprechend wird in jeder Iteration nur ein kleiner Teil der Daten gelernt und anschließend zu einem Großen zusammengesetzt. Allgemein ist der Learn++ eine Abwandlung von dem AdaBoost Algorithmus, welcher ebenso das inkrementelle Lernen implementiert. Der Unterschied ist jedoch, dass der Learn++ entwickelt wurde, um neue Klassen gut erkennen zu können, während der AdaBoost darauf ausgelegt ist die Genauigkeit der Klassifikation zu erhöhen. Damit der Learn++ inkrementell lernen kann, greift er auf Gewichte zurück. Jedem Datenpunkt wird ein einzelnes Gewicht zugewiesen, welches am Anfang initialisiert wird und in jedem Durchlauf aktualisiert wird. Wenn nach einem Durchgang ein Datenpunkt richtig erkannt worden ist, wird das Gewicht kleiner, sodass die Wahrscheinlichkeit kleiner wird, dass die Daten für das Training ausgewählt werden. Wie genau das in den Algorithmus eingebracht ist, ist in 2.2.2 genauer erläutert.

2.2.1 Input

Der Input für den Learn++ Algorithmus sind der Datensatz, ein schwacher Lerner und die Anzahl der Teildatensätze, in die der Datensatz aufgeteilt werden soll, sowie die Anzahl der Iterationen, die der Algorithmus durchlaufen soll. Der schwache Lerner ist meistens ein Entscheidungsbaum, welcher innerhalb des Algorithmus aufgebaut wird. Das Aufteilen des Datensatzes erfolgt, um das inkrementelle Lernen Schritt für Schritt darzustellen, dies gilt ebenfalls für die Anzahl der Iterationen, die auf einem Teildatensatz angewandt werden.
Der Learn++ Algorithmus besteht aus zwei Zählschleifen, wobei die erste die einzelnen Teildatensätze durchgeht und die zweite den eigentlichen Algorithmus und die Iterationen implementiert. Dabei entspricht K die Anzahl der Teildatensätze, k dem aktuellen Teildatensatz, T der Anzahl der Iterationen und t der aktuellen Iteration.

2.2.2 Zählschleife eins

Die erste Zählschleife geht die einzelnen Teildatensätze durch, das bedeutet, dass ein Datensatz der 10000 Daten enthält und in Teildatensätze mit 2000 Daten aufgeteilt wird, entsprechend fünf Durchgänge in der äußeren Zählschleife hat, damit jeder Teildatensatz einmal abgearbeitet wird. Weiterhin werden in der ersten Iteration, welche in der zweiten Zählschleife gezählt werden, die Startgewichte der einzelnen Daten festgelegt, diese entsprechen je Datenpunkt 1/m, wobei m die Anzahl aller Daten des Teildatensatzes ist.

2.2.3 Zählschleife zwei

Innerhalb der zweiten Schleife wird der eigentliche Algorithmus durchgeführt, dieser ist in sechs Schritte unterteilt.
Im ersten Schritt werden die Gewichte normalisiert, dazu werden die Gewichte durch die Summe aller Gewichte geteilt. Im zweiten Schritt wird ein zufälliges Test- und Trainingsset aus dem aktuellen Teildatensatz ausgewählt. Das Trainingsset wird nicht vollständig zufällig ausgewählt, sondern in Abhängigkeit der Gewichte, je größer ein Gewicht ist, desto wahrscheinlicher wird der Datenpunkt für das Trainingsset ausgewählt. Anschließend wird der schwache Lerner aufgerufen, dies stellt den dritten Schritt dar. Als Viertes wird die Hypothese von dem Trainings- und Testset aufgestellt. Anschließend wird der Fehler ε_t berechnet, dieser ist die Summe der Gewichte derjenigen Datenpunkte, welche von dem schwachen Lerner falsch klassifiziert worden sind. Ist der Fehler größer 0,5, was einer reinen Zufallsentscheidung bei binären Problemen entsprechen würde, wird die Hypothese verworfen und die aktuelle Iteration wird mit einem neuen Trainings- und Testset ab Schritt zwei wiederholt. Wenn der Fehler kleiner ist, wird der normalisierte Fehler β_t berechnet, mit $\beta_t = \varepsilon_t/(1-\varepsilon_t)$. Im fünften Schritt werden die zusammengesetzte Hypothese H_t und der zusammengesetzte Fehler E_t berechnet. Die zusammengesetzte Hypothese H_t berechnet sich mit einer gewichteten Größe, welche die Anzahl der Bilder der Klasse, die in der Hypothese am besten klassifiziert worden sind, und der Summe der Logarithmen der Kehrwerte des Fehlers β_t, als

Formel: $\sum log(1/\beta_t)$. Weiterhin wird der zusammengesetzte Fehler E_t ebenso wie in Schritt vier dargestellt berechnet. Der Durchgang wird genauso verworfen, wenn E_t größer als 0,5 ist, dies kann jedoch nur dann eintreten, wenn in der aktuellen Iteration neue Klassen eingeführt werden. Der sechste und letzte Schritt stellt das Herzstück des Algorithmus dar. Zuerst wird der zusammengesetzte normalisierte Fehler $B_t = E_t/(1 - E_t)$ berechnet. Damit werden die Gewichte des aktuellen Teildatensatzes aktualisiert. Wenn ein Datenpunkt richtig klassifiziert wurde, wird das Gewicht des Datenpunktes mit B_t multipliziert - durch die Daten und die Normalisierung ergibt sich, dass $0 < B_t < 1$ ist. Insgesamt wird das Gewicht also kleiner, weshalb eine erneute Auswahl in das nächste Trainingsset unwahrscheinlicher wird. Die falsch klassifizierten Datenpunkte behalten ihr Gewicht bei.
Nach der Aktualisierung der Gewichte wird der nächste Iterationsschritt eingeleitet oder der nächste Teildatensatz ausgewählt.

2.2.4 Abschluss

Nachdem der gesamte Datensatz mit allen Iterationen durchgelaufen ist, wird die finale Hypothese H_final berechnet. Diese berechnet sich aus dem Produkt der Anzahl der Datenpunkte, die über alle Iterationen am besten erkannt worden sind und der Summe aller Logarithmen der Kehrwerte des zusammengesetzten Fehlers B_t. Dabei gilt, je größer der Wert ist, desto besser hat der Algorithmus die Daten klassifiziert. [RP01]

3 Durchführung

3.1 Datensatz

Der Datensatz, der sowohl von dem Learn++ Algorithmus, als auch von dem CNN gelernt und getestet wird, ist der CIFAR10 Datensatz. Der Datensatz enthält 60000 Bilder aus zehn verschiedenen Klassen. Die Bilder haben eine Größe von 32x32 Pixeln, die geringe Größe hat den Vorteil, dass der Datensatz nicht zu groß wird, im Sinne von Speicherplatz und dass die Größe gleichzeitig noch ausreichend ist, damit der Mensch erkennen kann, was auf den Bildern dargestellt ist. Die zehn Klassen sind Flugzeuge, Autos, Vögel, Katzen, Hirsche, Hunde, Frösche, Pferde, Schiffe und LKW. Jede Klasse enthält jeweils 6000 Bilder, wobei 5000 zum Training und 1000 zum Testen gedacht sind, damit ergeben sich insgesamt 50000 Trainingsbilder und 10000 Testbilder, wenn der Tensorflow Import benutzt wird. Für unseren Learn++ Algorithmus ist der gesamte Datensatz in zwei Datensätze mit jeweils fünf Klassen aufgeteilt worden.

3.2 Umsetzung des Learn++ Algorithmus

Laden des Datensatzes

Der Learn++ Algorithmus wird darauf trainiert Bilder verschiedenen Typen zuzuordnen. Es gibt insgesamt 10 verschiedene Bildtypen: Flugzeuge, Autos, Vögel, Katzen, Rehe, Hunde, Frösche, Pferde, Schiffe und LKWs. Jede Bildklasse besteht aus 6000 Farbbildern der Größe 32x32. Der gesamte Datensatz hat eine Größe von 163MB.
Die numpy Funktion „load“ ist in der Lage zuvor abgespeicherte Variablen und Arrays in ein Programm zu laden. Um möglichst laufzeitschonend zu arbeiten wird diese Funktion verwendet, um den Datensatz zu initialisieren. Die Dateien, auf die „load“ zugreift müssen zunächst lokal gespeichert werden. Diese Dateien wurden auf GIThub bereitgestellt. Da GIThub nur Dateien von einer maximalen Größe von 20MB erlaubt, wurde der Datensatz in insgesamt 10 Teile unterteilt, bestehend aus jeweils einem Bildtyp.

Initialisierung der Daten

Nachdem der Datensatz in das Python Programm geladen wurde wird die Datenstruktur zunächst so verändert, dass es einfacher ist mit den Daten zu arbeiten. Zunächst werden die 10 Dateien, die in das Programm geladen wurden, zu zwei Arrays zusammengefasst. Der erste Array besteht aus den ersten 5 Bildtypen, und der zweite aus den verbliebenen 5 Bildtypen. Diese Unterteilung wurde vorgenommen, weil getestet wird, wie gut der Algorithmus sich an neue Bildtypen adaptieren kann. Um dies zu testen trainiert der Algorithmus zunächst ausschließlich mit den ersten 5 Bildtypen und anschließend mit den verbliebenen. Daher wird der im Anschluss beschriebene Code als erstes auf das erste Array angewendet und danach auf das zweite.
Jedem Bild werden zwei weitere Werte zugewiesen. Welchem Bildtypen sie zugehören und ein Wert für die Gewichtung. Der Bildtyp ist wird gespeichert, damit zu jedem Punkt jedes Bild eindeutig zu seinem Bildtyp zugeordnet werden kann. Das Gewicht wird später verwendet um zu bestimmen welche Bilder zum Trainieren verwendet werden.
Im nächsten Schritt werden alle Daten in K gleichgroße Gruppen unterteilt. Für diesen Datensatz hat sich K=5 als geeignet herausgestellt. Jede Gruppe besteht in dem Fall aus insgesamt 6000 Bildern, 1200 von jedem Bildtyp. Der Learn++ Algorithmus wird der Reihe nach mit allen Gruppen trainiert. Jede Gruppe wird dabei T Mal hintereinander verwendet. Nach 20 Durchläufen haben sich die Ergebnisse nur noch

unwesentlich verbessert. Aus dem Grund wurde T=20 gewählt.

Bilden von Datenpaketen

In jedem Durchlauf von jeder Gruppe werden als erstes eine Trainingsgruppe und eine Testgruppe ausgewählt. Wie wahrscheinlich es ist, dass ein Bild ausgewählt wird hängt von dem aktuellen Gewicht ab, welches das entsprechende Bild hat. Desto größer das Gewicht, desto wahrscheinlicher ist es, dass dieses ausgewählt wird. Um dies in dem Programm umzusetzen werden zunächst Zufallszahlen im Bereich von 0 bis 1 erzeugt. Die Anzahl der Zufallszahlen entspricht der gewünschten Größe der jeweiligen Gruppe. Für diesen Datensatz eignet sich eine Trainingsgruppengröße von 3000 und eine Testgruppengröße von 1500. Als nächstes werden die generierten Zufallszahlen der Größe nach sortiert. Im letzten Schritt wird der reihe Nach jedes Bild der Gruppe durchlaufen und die Summe von allen bereits durchlaufenen Bildern gebildet. Die Summe aller durchlaufenen Gewichte wird mit der aktuell kleinsten Zufallszahl verglichen. Wenn die Summe größer ist als die kleinste Zufallszahl wird das Bild von dem aktuellen Durchlauf der Trainings oder entsprechen der Testgruppe zugewiesen. Des Weiteren wird diese Zufallszahl verworfen und die nächstgrößere Zahl ist die neue kleinste Zufallszahl. Die Summe von allen Gewichten hat den Wert 1. Aus dem Grund wurden die Zufallszahlen in dem Wertebereich von 0 bis 1 gewählt.
Ein Problem bei diesem Verfahren ist, dass es passieren kann, dass zwei oder mehr Zufallszahlen im Wertebereich von einem Gewicht liegen. In dem Fall wird das jeweils nächste Bild zusätzlich ausgewählt unabhängig von dessen Gewicht. Dies hat den Effekt, dass wenn dieser Spezialfall auftritt, es wesentlich wahrscheinlicher ist, dass ein Bild mit einem kleinem Gewicht ausgewählt wird. Dieser Effekt ist nicht erwünscht.

Trainieren des Learn++ Algorithmus

Mit den gebildeten Trainings und Testgruppen wird der CNN trainiert und getestet. Der Algorithmus Learn++ ist darauf spezialisiert mit schwachen Lernern gute Ergebnisse zu erreichen. Da das CNN, welches verwendet wurde, ein sehr starker lernen ist wurde dieses künstlich geschwächt. Die Anzahl der Bäume sowie die Anzahl der Durchläufe wurde verringert, sowie 12Reg erhöht. Von der Testgruppe wird ermittelt, welche Klasse am besten zugeordnet wurde, wie häufig diese richtig zugeordnet wurde und welche Bilder richtig und welche falsch zugeordnet wurden.
Um zu überprüfen, ob das Ergebnis von dem CNN gut genug ist, um dieses weiter zu verwenden, werden von allen Bildern, die falsch zugeordnet wurden, die Gewichte aufsummiert. Diese Summe wird mit einem Kontrollwert verglichen. Der Kontrollwert entspricht 0.5, was 50% von der Summe von allen Gewichten in der Gruppe entspricht, multipliziert mit der relativen Größe der Testgruppe (Anzahl von Daten/Daten in der Testgruppe). Wenn die Summe der Gewichte größer als der Kontrollwert ist, wird der Lernerfolg von dem CNN als unzureichend eingestuft und die Ergebnisse werden verworfen und neue Trainings und Testgruppen werden ausgewählt.
Falls die Bedingung erfüllt wurde wird im nächsten Schritt das Ergebnis von dem CNN ein weiteres Mal überprüft. Die zusammengesetzte Hypothese der aktuellen Iteration wird berechnet. Mit Hilfe von dieser sollte E_t ausgerechnet werden, was allerdings in dem Algorithmus noch nicht korrekt implementiert ist. Wenn auch diese Bedingung erfüllt ist wird das Trainieren von dem CNN erfolgreich. In dem Fall wird der normalisierte Fehler berechnet. Mithilfe der normalisierten Hypothese können die Gewichte der einzelnen Bilder angepasst werden. Die Gewichte von allen Bildern, die zuvor von dem CNN richtig zugeordnet wurden, werden mit dem normalisierten Fehler multipliert. Da der normalisierte Fehler immer kleiner als 1 ist hat dies zur Folge, dass die Gewichte von diesen Bildern kleiner werden und die Wahrscheinlichkeit, dass diese im nächsten durchlauf ausgewählt werden sinkt. Anschließend werden die Gewichte normalisiert, sodass die Summe aller Gewichte wieder 1 ergibt.

Bewertung des Ergebnisses

Nachdem der CNN mit jeder Gruppe T mal trainiert wurde, wird als letztes die finale Hypothese berechnet. Diese bewertet das Ergebnis des Algorithmus. Der gesamte Vorgang wird mit dem zweiten Satz von Bildern wiederholt. Damit ist der Lernvorgang abgeschlossen.

Anwendung des trainierten Algorithmus

Wenn mithilfe des fertig trainierten Learn++ Algorithmus ein Bild einer Kategorie zugewiesen werden soll wird dieses jedem zuvor generierten neuronalen Netz übergeben. Jedes neuronale Netz berechnet eine Wahrscheinlichkeit zu der dieses Bild einem bestimmten Bildtypen zugehört. Die Wahrscheinlichkeiten für jeden Bildtypen von jedem Netz werden aufsummiert. Der Bildtyp mit der höchsten Gesamtwahrscheinlichkeit wird als Endprognose ausgegeben.
Jedes neurale Netz wurde nur mit 5 der insgesammt 10 verschiedenen Bildklassen trainiert. Wenn das Bild nicht in einer der Klassen ist, mit der das Netz trainiert wurde kann es das Bild nicht korrekt zuweisen. Allerdings teilen sich alle Fehlprognosen auf die 9 falschen Bildklassen auf. Aus dem Grund ist in den meisten Fällen die Bildklassen mit der höchsten Wahrscheinlichkeit auch die richtige Klasse.

3.3 Neuronales Netz

Eine weitere Aufgabe war es den Datensatz mit einem klassischen neuronalen Netz zu testen und anschließend mit dem Learn++ zu vergleichen. In unserem Projekt wurde ein Convolutional Neural Network, kurz CNN, verwendet, welches sich auf Grund seiner Faltungen gut zur Klassifizierung von Bildern eignet. Das benutze CNN ist das von Professor Frochte in der Vorlesung vorgestellte Netz, der Quellcode findet sich auf folgender Webseite unter dem Punkt Sourcecode. Zunächst werden die Daten und die für das neuronale Netz benötigten Bibliotheken geladen, der CIFAR10 Datensatz wird dabei als NumPy Array, aufgeteilt in Test- und Trainingsmenge, gespeichert. Der Datensatz liegt im RGB Format vor, weshalb die Werte durch 255 geteilt werden, um diese auf den Wertebereich [0,1] zu skalieren, damit die Verarbeitung für das Netz einfacher wird. Im nächsten Schritt wird eine L2-Regularisierung eingefügt, um das Overfitting zu vermeiden. Nachdem das Sequential Modell erstellt worden ist, werden jeweils zwei Faltungs- und Poolingslayer abwechselnd eingefügt. Die ersten beiden werden mit 32 Filtern versehen, die letzten beiden mit 64. Nachdem die Bilder gefaltet wurden, werden diese mit dem Flatten für das eigentliche neurale Netz vorbereitet. Dieses besteht aus 512 Eingangslayern, 256 Hiddenlayern und entsprechend der zehn Klassen zehn Ausgangslayer. Anders, als in der Vorlage werden anschließend die Gewichte geladen, mit denen die höchste Genauigkeit erreicht wurde, für die Gewichte wurde das Netz mit 20 Epochen trainiert. Als letztes wird die Genauigkeit ausgegeben, mit welcher das Netz die Testbilder den Klassen zugeordnet hat.[Fro21]
Um das Ergebnis mit dem vom Learn++ vergleichen zu können, wurde zum einen die Genauigkeit auf dem Testset berechnet, als auch ausgegeben, wie gut einzelne Klassen erkannt worden sind und bei welchen Klassen häufig Verwechselungen stattfinden.

4 Ergebnisse

4.1 Ergebnisse Learn++

Der Learn++ Algorithmus wurde wie zuvor beschrieben mit den Bildern trainiert. Dabei wurde eine Vielzahl von neuronalen Netzen erstellt. Jedes Bild wird von allen Netzen bewertet und das Ergebnis, welches von allen Netzen kombiniert am wahrscheinlichsten ist, wird als Endergebnis angenommen. Mit diesem Verfahren wurden die selben Bilder, wie beim testen des CNN, ausgewertet. Der Algorithmus konnte eine Genauigkeit von 49% erreichen. Für eine erweiterte Analyse wurde eine Tabelle generiert, in der dargestellt wird, welcher Bildtyp mit welchem anderen verwechselt wurde. Die Zahl in jeder Zelle beschreibt wie häufig ein Bild von dem Bildtyp der Spalte durch den Algorithmus dem Bildtyp der Zeile zugeordnet wurde.

	Flugzeug	Autos	Vögel	Katzen	Hirsche	Hunde	Frösche	Pferde	Boote	LKW
Flugzeug	345	20	28	8	24	9	6	20	83	58
Autos	41	695	13	33	12	8	31	18	107	280
Vögel	14	0	174	20	24	37	16	11	1	5
Katzen	15	10	47	360	33	187	92	86	19	22
Hirsche	3	3	52	31	296	44	33	60	3	6
Hunde	44	20	200	273	115	495	39	76	24	17
Frösche	40	30	228	132	265	75	728	50	15	26
Pferde	67	27	116	74	171	98	19	629	16	33
Boote	381	59	117	40	45	34	18	16	693	55
LKW	50	136	25	29	15	13	18	34	39	498

Abbildung 4.1: Zuordnungsmatrix CNN

Der Algorithmus wurde zuerst mit den ersten 5 Bildtypen trainiert, und anschließend mit den verbliebenen 5. Es zeigt sich, dass der Algorithmus den zweiten Datensatz wesentlich besser verarbeiten als den ersten. Bewertet man die beiden Gruppen seperat ergibt sich eine Genauigkeit von 37,4% für die erste Gruppe und 60,9% für die restlichen Bildtypen. Des weiteren fällt auf, dass ein Bild meistens einem Bildtypen der anderen Gruppe zugeordnet wurde, wenn das Ergebnis falsch war. Die meisten Fehler kommen zustande, weil die neuronalen Netze, welche mit den anderen Bildgruppen trainiert wurden, alle ein Bild der gleichen Falschen Gruppe zuordnen. Werden nur die Ergebnisse von den Netzen betrachtet, die unteranderen mit dem richtigen Bildtyp trainiert wurden, erhält man für die erste Bildgruppe eine Genauigkeit von 81,3% für die erste und 81,0 für die zweite.
Wenn die Bildtypen anders aufgeteilt werden, sodass in der ersten Datengruppe andere Bildtypen sind, verändert sich auch das Ergebnis. In den meisten Fällen wird eine Datengruppe besser erkannt, als die andere. Der Effekt ist allerdings in den meisten Kombinationen weniger extrem.

4.2 Ergebnisse CNN

Nachdem das CNN, wie in 3.3 beschrieben, mit den 50000 Bildern in 20 Epochen gelernt wurde, wurde bei dem Test der 10000 Bilder eine Genauigkeit von 74,59% erreicht. Um die Ergebnisse noch genauer miteinander vergleichen zu können, haben wir uns genau ausgegeben, welche Bilder besser und welche schlechter klassifiziert worden sind, dies ist in der folgenden Tabelle dargestellt. In der ersten Zeile sind

die Klassen, welchen die Bilder angehören und in den Spalten, als was die Bilder klassifiziert worden sind.

	Flugzeug	Autos	Vögel	Katzen	Hirsche	Hunde	Frösche	Pferde	Boote	LKW
Flugzeug	787	21	89	43	32	24	9	31	49	44
Autos	13	853	6	11	3	7	3	0	8	53
Vögel	26	1	601	57	66	41	26	20	3	3
Katzen	8	4	34	484	50	107	30	25	4	4
Hirsche	10	0	57	43	660	35	21	35	1	1
Hunde	4	1	58	159	31	652	14	57	1	3
Frösche	8	5	87	95	69	39	868	4	4	7
Pferde	7	2	32	40	73	63	6	802	2	6
Boote	102	47	24	43	15	21	19	8	911	38
LKW	35	66	12	25	1	11	4	18	17	841

Abbildung 4.2: Zuordnungsmatrix CNN

Die Klasse, welche am besten erkannt worden ist, sind die Boote. Dabei sind 911 von 1000 Booten richtig erkannt worden, dies entspricht einer Genauigkeit von 91,1%. Am schlechtesten wurden die Katzen erkannt, dort gab es nur eine Genauigkeit von 48,4%. Die Katzen, welche nicht richtig zugeordnet worden sind, wurden am häufigsten als Hunde erkannt. Dies war bei 159 Bildern der Fall. Umgekehrt wurden auch 107 Hunde als Katzen klassifiziert, das zeigt, dass die Bilderkennung gerade bei Tieren die sich ähnlich sehen an die Grenzen kommt. Weiterhin sind auch 102 Flugzeuge als Boote erkannt worden. Die Ursache hierfür könnte ein ähnlicher Hintergrund darstellen, da sowohl der Himmel bei Flugzeugen blau ist, als auch das Wasser bei den Booten.

5 Fazit und Ausblick

5.1 Fazit

Das CNN funktioniert genauso, wie erwartet. In mehreren Tests hat sich ergeben, dass man für ein klassisches CNN, das mit dem CIFAR10 getest wird, eine Genauigkeit von ca. 75% erwarten kann.[Fro21]
In unserem Versuch wurde eine Genauigkeit von 74,59% erreicht. Wie vermutet kommt es zu Erkennungsschwierigkeiten bei Hunden und Katzen, sowie bei Booten und Flugzeugen.
Mit einer Genauigkeit von 49% ist das Ergebnis des Learn++ Ansatzes deutlich schlechter. Wenn der Algorithmus auf 5 Bildtypen angewendet wird, erreicht dieser eine Genauigkeit von 81%. Durch das nachträgliche erlernen der verbliebenen Bildtypen sinkt die Genauigkeit um weitere 32%. Es konnte bewiesen werden, dass mithilfe des Learn++ Ansatzes es möglich ist, neue Bildtypen zu erlernen, ohne ein katastrophales Vergessen des zuvor erlernten. Die Zuverlässigkeit des Algorithmus ist geringer als erhofft.

5.2 Ausblick

Die Berechnung des Fehlers mithilfe der zusammengesetzten Hypothese wurde noch nicht korrekt implementiert.
Des weiteren könnte der Algorithmus deutlich verbessert werden, wenn eine Möglichkeit gefunden wird, durch die die Ergebnisse von neuronalen Netzen, welche das richtige Ergebnis nicht finden können, da sie nicht auf diesen Bildtypen trainiert wurden, weniger stark ins Gewicht fallen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AAR16] Andrei A. Rusu, Guillaume Desjardins Hubert Soyer James Kirkpatrick Koray Kavukcuoglu Razvan Pascanu Raia H. Neil C. Rabinowitz R. Neil C. Rabinowitz: Progressive Neural Networks. In: *DeepMind London, UK* (2016), S. 2

[CL18] Chen, Z. ; Liu, B.: Continual Learning and Catastrophic Forgetting. In: *. Morgan Claypool Publishers* (2018), S. 56–61

[Fre99] French, Robert: Catastrophic forgetting in connectionist networks. In: *French national centre for scientific reaserch* (1999), S. 2

[Fro21] Frochte, Jörg: *Maschinelles Lernen: Grundlagen und Algorithmen in Python.* Carl Hanser Verlag München, 2021. – ISBN 978–3–446–46144–4

[GIP19] German I. Parisi, Jose L. Part Christopher Kanan Stefan W. Ronald Kemker K. Ronald Kemker: Continual Lifelong Learning with Neural Networks: A Review. In: *C1Knowledge Technology, Department of Informatics, Universitat Hamburg, Germany ¨Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, NY, USA Department of Computer Science, Heriot-Watt University, Edinburgh Centre for Robotics, Scotland, UK* (2019), S. 2,3,4,5

[Kar] Karin: *Neuronale Netze und die Hebbsche Lern regel.* url=https://www.wipub.net/neuronale-netze-und-die-hebbsche-lernregel/, . – Zugriff: 26.03.2021

[RP01] Robi Polikar, Satish S. Udpa Vasant H. Lalita Udpa U. Lalita Udpa: Learn++: An incremental learning algorithm for supervised neural networks. In: *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)* (2001), S. 499–501

[Sch] Schipek, Peter: *So lernt das Gehirn.* url=https://www.lernwelt.art, . – Zugriff: 26.03.2021

[Ull] Ullmann, Dr. E.: *Lernen aus neurobiologischer Perspektive.* url=https://www.uni-wuerzburg.de/fileadmin/06000060/04$_{F}ort$ $-_{u}$ $nd_{W}eiterbildungen_{L}ehrkraefte/Herbsttagungen/Herbsttagung_{2}016/20161006_{W}S_{0}4_{N}eurobiologie.pdf$, .– $-Zugriff$: 27.03.2021

[YLM20] Yong Luo, Wenchao B. Liancheng Yin Y. Liancheng Yin ; Mao, Keming: An Appraisal of Incremental Learning Methods. In: *College of Software, Northeastern University, Shenyang 110004, China; 1971141@stu.neu.edu.cn (Y.L.); 1971179@stu.neu.edu.cn (L.Y.); 20185085@stu.neu.edu.cn (W.B.)* (2020), S. 3