|  |  |  |
| --- | --- | --- |
| **MA 3062** | | |
| Untersuchung und prototypische Umsetzung eines Lifelong Deep Neural Network Algorithmus | | |
| **Simon Kamm** | | |
|  | | |
| **Pflichtenheft** | | |
|  | 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 |  | 02.05.19 | in Bearb. | Erstellung |
| 1.0 | Kamm | Ms | 07.05.19 | Vorgelegt | Finalisierung Pflichtenheft |
| 1.0 | Kamm | Ms | 13.05.19 | Akzeptiert | Durchsicht durch Ms |

# Inhaltsverzeichnis

0 Inhaltsverzeichnis 2

1 Zielbestimmung 4

1.1 Musskriterien 4

1.2 Wunschkriterien 4

1.3 Abgrenzungskriterien 4

2 Einsatz 4

2.1 Anwendungsbereiche 4

2.2 Zielgruppen 4

3 Anforderungen an die Konzeption 4

4 Qualitäts-Zielbestimmung 4

5 Durchführung 5

6 Prototyp\* 5

6.1 Umgebung 6

6.1.1 Software 6

6.1.2 Hardware 6

6.1.3 Systemschnittstellen 6

6.2 Qualitäts-Zielbestimmung für den Prototyp 6

6.3 Entwicklungs-Umgebung 8

6.3.1 Software 8

6.3.2 Hardware 8

7 Globale Evaluierungsmethoden 8

8 Ergänzungen 8

# Zielbestimmung

Im Rahmen der Arbeit wird das Konzept „Lifelong Deep Neural Network“ hinsichtlich seiner Funktionalität und Anwendbarkeit auf andere Aufgabengebiete analysiert.

## Musskriterien

Das in [1] vorgestellte Konzept von „Lifelong Deep Neural Network Algorithms“ (L DNN A) muss auf seine Umsetzbarkeit/Anwendbarkeit untersucht werden.

Vorhandene Beschreibungen des genannten Konzepts müssen untersucht und analysiert werden. Die Ergebnisse dieser Recherche müssen dann auf ihre Plausibilität überprüft sowie die Funktionsweise und der Aufbau von L DNN A anschaulich dargestellt werden.

Es wird eine prototypische Software implementiert, die es erlaubt auf Basis frei verfügbarer Bilddatensätze die Ansätze und Funktionsweise zu analysieren. Die Anwendung für die prototypische Implementierung ist die Objekterkennung/-klassifizierung. Generell kann dabei in die bereits genannten Themen kontinuierliches Lernen und verteiltes Lernen unterteilt werden.

Untersuchung ob „continual/lifelong Learning“ verwendbar ist mit den beschriebenen Konzepten in [1]. Dafür wird die Performanz dieses Ansatzes hinsichtlich seiner Fähigkeit zum kontinuierlichen erlernen neuer Objektklassen untersucht. Zum Vergleich werden bekannte Metriken genutzt sowie aktuelle „state-of-the-art“ Ansätze für das kontinuierliche Lernen als Vergleich herangezogen.

Als zweiter Teil müssen die beschriebenen verteilten Lernansätze untersucht werden.

Insgesamt muss untersucht werden, ob die in [1] beschriebenen Ansätze umsetzbar sind und wo eventuell andere Ansätze ebenfalls genutzt werden könnten bzw. vielversprechend wären oder es gegebenenfalls Lücken in der beschriebenen Methodik gibt.

## Wunschkriterien

Optional soll in dieser Arbeit ein Demonstrator aufgebaut werden, der die untersuchten Fähigkeiten des Ansatzes für einen Anwendungsfall anschaulich darstellt. Beispielhaft kann das eine Objekterkennung auf Basis von Live-Bildern sein, bei der neue Objektklassen erlernt werde.

## Abgrenzungskriterien

In dieser Arbeit werden keine grundlegenden Untersuchungen bezüglich Bildverarbeitung/Objekterkennung mithilfe „Deep Neural Networks“ (DNNs) angestellt.

Für das Konzept gibt es viele mögliche Anwendungsfälle. Im Rahmen dieser Arbeit wird der Anwendungsfall der Objekterkennung/-klassifizierung auf Basis frei verfügbarer Bilddatensätze untersucht, um das Potenzial des Konzepts prinzipiell zu untersuchen.

Der optionale Prototyp dient der Veranschaulichung des Konzepts. Es geht nicht darum einen möglichst perfekten HW-Prototypen aufzubauen, d.h. es wird kein aufwändiger und komplexer HW-Aufbau durchgeführt, sondern eine einfache Version zur anschaulichen Darstellung.

# Einsatz

## Anwendungsbereiche

Das in [1] dargestellte Konzept wird sehr generell beschrieben, und soll in der Theorie auf alle Anwendungsbereiche angepasst werden können, in denen weiterlernende oder verteilte DNNs eingesetzt werden können, z.B. Infrastrukturüberwachung mithilfe von Drohnenflotten oder „predictive Maintenance“ für Produktionsanlagen, bei der gleiche Maschinen an mehreren Standorten verteilt sind. Dadurch können z.B. neue, bisher unbekannte Fehler, die zu einem Maschinenabschalten an Maschine A geführt haben frühzeitig erkannt werden und dieses Wissen an Maschine B übermittelt werden. Maschine B kann dann einen möglichen Fehler frühzeitig erkennen und damit eine vorsorgliche Wartung („predictive Maintenance“) der Maschine anfordern.

Im Rahmen dieser Arbeit wird die generelle Umsetzbarkeit mithilfe dem Anwendungsfall der Objekterkennung/-klassifizierung analysiert. Frei verfügbare Bilddatensätze dienen dabei als Datenbasis.

## Zielgruppen

Als Zielgruppen können generell alle Firmen/Institute/o.ä. gesehen werden, welche viele gleiche Systeme im Einsatz haben. Dadurch lässt sich der Vorteil von L DNN A hervorragend nutzen, da durch die sehr große Anzahl System eine riesige Menge an Daten gesammelt werden kann, und die einzelnen DNNs sich durch stetig weiter verbessern können. Zudem können die einzelnen DNNs durch Austausch ihr neu erzieltes Wissen teilen. Durch den beschriebenen Ansatz soll es möglich sein auf den jeweiligen Endgeräten direkt zu lernen, wodurch auch kein großer Datentransfer/-speicher notwendig wäre in diesem Fall. In dieser Arbeit wird allerdings nur eine prototypische Implementierung umgesetzt. Optional wird ein Demonstrator aufgebaut. Somit kann als Zielgruppe dieser Arbeit Studenten, Institutsmitarbeiter oder technikaufgeschlossene Personen gesehen werden. Diesen kann mit einem optionalen Demonstrator an öffentlichen Veranstaltungen (z.B. Tag der Wissenschaft) das Model sowie die Fähigkeit des Ansatzes gezeigt werden.

# Anforderungen an die Konzeption

|  |  |
| --- | --- |
| **/PA10/** | **Recherche und Analyse der verfügbaren Beschreibung von L DNN A (/LA10/)** |
| /PA11/ | Grundlegende Literaturrecherche zu den Themen „Deep Neural Networks“ und „Deep Learning“ |
| /PA12/ | Allgemeine Literaturrecherche zu den Themen „continual“ und „distributed“ Learning mit Fokus auf aktuelle Architekturen zu den Themen |
| /PA13/ | Detaillierte Analyse der verfügbaren Beschreibungen von L DNN A |
| **/PA20/** | **Plausibilitätprüfung der Ergebnisse von /PA10/ und anschauliche Darstellung des Aufbaus und der Funktionsweise von L DNN A (siehe /LA20/)** |
| /PA21/ | Bewertung der Recherche- und Analyseergebnisse |
| /PA22/ | Anschauliche Darstellung des Aufbaus und der Funktionsweise von L DNN A sowie Vergleich zu bisherigen Architekturen |
| **/PA30/** | **Prototypische Implementierung des L DNN A zur Evaluierung auf Basis eines frei verfügbaren Bild-Datensatzes (siehe /LA30/)** |
| /PA31/ | Auswahl einer geeigneten Programmiersprache (z.B. TensorFlow) |
| /PA32/ | Es muss ein separates, langsam lernendes Modul zur Feature-Extraktion implementiert werden |
| /PA33/ | Es muss ein separates, schnell lernendes Modul zur Objektklassifizierung implementiert werden auf Basis der eingespeisten Daten |
| /PA34/ | Die implementierten Module müssen kombiniert werden und eine „Pipeline“ zum korrekten Kontrollfluss der Daten implementiert werden |
| **/PA40/** | **Evaluierung der Ergebnisse von /PA30/ anhand eines frei verfügbaren Bilddatensatzes (siehe /LA40/)** |
| /PA41/ | Evaluierung der Ergebnisse von /PA30/ hinsichtlich des Aspektes „continual learning“ anhand eines frei verfügbaren Bilddatensatzes |
| /PA42/ | Evaluierung der Ergebnisse von /PA30/ hinsichtlich des Aspektes „distributed learning“ anhand eines frei verfügbaren Bilddatensatzes |
| **/PA50/** | **Diskussion der Ergebnisse von /PA41/ und /PA42/ und Bewertung des Potentials von L DNN A im Allgemeinen und hinsichtlich der Claims in [1] im Besonderen (siehe /LA50/)** |
| /PA51/ | Einordnen der Ergebnisse in Vergleich zu anderen (vergleichbaren) aktuellen Architekturen |
| /PA52/ | Bewertung der Claims in [1] auf Basis der erzielten Resultate |
| /PA53/ | Aussage über Weiterverwendbarkeit von einzelnen Punkten oder des gesamten Ansatzes |
| **/PA60/** | **Optional: Erweiterung der prototypischen Implementierung um die Möglichkeit einer Live-Bild-Analyse inkl. Der dafür notwendigen Hardware-Anordnung zur anschaulichen Demonstration des Konzeptes sowie dessen Potenzial (siehe /LA60/)** |

# Qualitäts-Zielbestimmung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | sehr hoch | hoch | normal | nicht relevant |
| Theorie | **x** |  |  |  |
| Allgemeingültigkeit | x |  |  |  |
| Konsistenz | x |  |  |  |
| Funktionalität | **x** |  |  |  |
| Richtigkeit | x |  |  |  |
| Anwendbarkeit | x |  |  |  |
| Sicherheit |  |  |  | **x** |
| Benutzbarkeit |  | **x** |  |  |
| Verständlichkeit |  | x |  |  |
| Erlernbarkeit |  |  | x |  |
| Änderbarkeit |  |  | **x** |  |
| Analysierbarkeit |  |  | x |  |
| Modifizierbarkeit |  |  | x |  |

# Durchführung

Die Arbeit wird nach dem IAS-Vorgehensmodell (Modell für konzeptionelle Arbeiten) durchgeführt werden.

Der Stand der Arbeit und die Ergebnisse werden in regelmäßigen Abständen (ca. alle 2 Wochen) mit den Betreuern diskutiert.

Bei der Durchführung der Arbeit und der Anfertigung der Ausarbeitung werden die Richtlinien des IAS beachtet.

# Prototyp

In diesem Kapitel wird auf die prototypische Implementierung eingegangen. Die grundlegenden Systemschnittstellen sowie Qualitätszielbestimmungen für den optionalen Demonstrator sind identisch zu dem Software-Prototyp. Zudem soll die entwickelte Software wiederverwendet werden. Lediglich die genutzte Hardware unterscheidet sich und wird bei Umsetzung separat beschrieben, jedoch zum aktuellen Zeitpunkt des Projektes noch nicht näher definiert.

Die prototypische Implementierung soll das Potenzial von L DNN A auf Basis von frei verfügbaren Bilddatensätzen untersuchen. Mit der prototypischen Implementierung sollen verschiedene definierte Experimente durchgeführt werden um die Claims in [1] und [2] untersuchen und bewerten zu können.

## Umgebung

### Software

Die prototypische Implementierung wird in der Python-basierten Programmiersprache *TensorFlow* implementiert. Dafür werden verschiedene Python- und TensorFlow- interne Pakete und Plug-Ins genutzt, um eine effiziente Programmierung und Ausführung des Codes zu gewährleisten.

### Hardware

Die prototypische Implementierung findet rein in Software statt. Die prototypische Implementierung lässt sich auf einem normalen Windows PC durchführen

### Systemschnittstellen

Verschiedene Schnittstellen des Systems sind notwendig. Generell lässt sich der SW-Prototyp von einem Windows PC aus bedienen:

* Eine Schnittstelle ist für das Einlesen der Daten notwendig, wo die Bilddaten in das System eingespeist werden.
* Output-Schnittstelle für die finale Bewertung/Klassifizierung der Input-Daten zur Berechnung von geeigneten Metriken
* Schnittstelle zum möglichen Austausch von Gewichten der DNNs, somit zum Austausch von neu erlerntem Wissen zwischen verschiedenen (verteilten) DNNs
* Interne Schnittstelle für den Austausch von Informationen von Modul A nach Modul B (Details siehe [1])

## Qualitäts-Zielbestimmung für den Prototyp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | sehr hoch | hoch | normal | nicht relevant |
| Funktionalität | **x** |  |  |  |
| Richtigkeit | x |  |  |  |
| Sicherheit (Security) |  |  |  | x |
| Interoperabilität | x |  |  |  |
| Zuverlässigkeit |  |  |  | **x** |
| Reife |  |  |  | x |
| Fehlertoleranz |  |  |  | x |
| Wiederherstellbarkeit |  |  |  | x |
| Sicherheit (Safety) |  |  |  | **x** |
| Benutzbarkeit |  | **x** |  |  |
| Verständlichkeit | x |  |  |  |
| Erlernbarkeit |  |  | x |  |
| Bedienbarkeit |  |  | x |  |
| Effizienz |  | **x** |  |  |
| Zeitverhalten |  | x |  |  |
| Verbrauchsverhalten |  | x |  |  |
| Änderbarkeit |  |  | **x** |  |
| Analysierbarkeit |  |  | x |  |
| Modifizierbarkeit |  |  | x |  |
| Übertragbarkeit |  |  | **x** |  |

## Entwicklungs-Umgebung

### Software

Die entwickelte Software wird in Python-basierten APIs geschrieben, weshalb als Software Entwicklungsumgebung *Pycharm* genutzt wird.

### Hardware

Im generellen wird keine spezielle Hardwareumgebung genutzt. Eventuell wird der vorhandene GPU-Server genutzt, um ein schnelleres Training der DNNs zu ermöglichen.

Für den optionalen Prototypen wird ein handelsüblicher Mikrocontroller genutzt, der zum jetzigen Zeitpunkt noch nicht festgelegt ist.

# Globale Evaluierungsmethoden

Die prototypische Software-Implementierung wird gegen die im Pflichtenheft im Kapitel 4 und 6.2 definierten Anforderungen evaluiert. Im Rahmen der Arbeit liegt der Fokus auf dem Potenzial der Architektur. Beispielhaft wird dafür der Anwendungsfall der Objekterkennung/-klassifizierung untersucht.

Zur Evaluierung werden typischen Metriken für den Anwendungsfall der Klassifizierung, wie z.B. „True-Positive Rate“, genutzt.

Zusätzlich können spezifische Metriken für die einzelnen Kategorien des „continual Learnings“ und des „distributed Learnings“ eingesetzt werden um eine detailliertere Analyse zu ermöglichen.

# Ergänzungen

Keine.

# Literatur

[1] **Luciw, M. et al:** *Systems and methods to enable continual, memory-bounded learning in artificial intelligence and deep learning continuously operating applications across networked compute edges,* United States Patent Application Publication No. US 2018/0330238 A1, 2018

[2] **Neurala Inc.:** *Lifelong-Deep Neural Networks – Tech Summary*, White Paper, Boston, 2018, verfügbar unter https://info.neurala.com/hubfs/docs/

Neurala\_LifelongDNNWhitepaper.pdf, zuletzt aufgerufen am 03.05.2019