**Pflichtenheft: Bilderkennung**

# Inhaltsverzeichnis

1 Inhaltsverzeichnis 1

2 Zielbestimmung 2

2.1 Muss-Kriterien 2

2.2 Soll-Kriterien 3

2.3 Kann-Kriterien 3

2.4 Abgrenzungskriterien 4

3 Produkteinsatz 4

3.1 Anwendungsbereiche 4

3.2 Zielgruppe 5

3.3 Betriebsbedingungen 5

4 Produktumgebung 5

4.1 Software 5

4.2 Hardware 5

5 Produktfunktionen 6

6 Qualitätsbestimmungen 7

7 Testszenarien und Testfälle 8

8 Schnittstellen 9

8.1 Bildinformationen 9

8.1.1 Import 9

8.1.2 Export 9

8.2 Intelligenz 9

8.2.1 Import 9

8.2.2 Export 10

9 Entwicklungsumgebung 10

9.1 Software 10

9.2 Hardware 10

# Zielbestimmung

Das Ziel der Studienarbeit ist eine wissenschaftliche Diskussion über einsetzbare Verfahren zur Bildanalyse um Objekte innerhalb eines Bildes zu Erkennen und Kategorisieren. Zusätzlich soll eine Anwendung als Resultat dieser Analogie entwickelt werden, welche eine Server-Einbettung ermöglicht um eine solche Objekterkennung innerhalb einer Serveranwendung umsetzen zu können.

Muss-Kriterien sind auf jeden Fall verpflichtend, Soll-Kriterien sind so weit wie möglich zu implementieren, Kann-Kriterien sind optional und werden implementiert, wenn noch Zeit übrig ist.

## Muss-Kriterien

* Die Anwendung enthält ein funktionsfähiges Neuronales Netz.
* Dieses neuronale Netz ermöglicht die Erkennung und Lokalisierung von Objekten innerhalb eines Bildes. Aufgrund dieser Funktionalität wird dieses neuronale Netz in folgenden als *object recognition application* bezeichnet.
* Das neuronale Netz enthält eine Eingabe- und Ausgabe-Schnittstelle für Bilder.
* Die Architektur des neuronalen Netzes muss dokumentiert sein.
* Das neuronale Netz kann mit Hilfe von „supervised-training“ trainiert werden.
* Die Anwendung muss mit einer Datei zur grundlegenden vortrainierten Intelligenz ausgeliefert werden.
* Der Algorithmus zum Training (supervised) der Daten soll das ihm gegebene Trainingsmaterial maximal ausnutzen.
* Die Anwendung soll in Python2 programmiert werden. Aufgrund der fehlenden Komptabilität der offen verfügbaren Bibliotheken sei hierbei auch höhere Versionen 2.x, nicht jedoch Python der dritten Generation miteinbegriffen. Für eine zukünftige Migration der Anwendung zur dritten Generation soll jedoch eine mögliche Aufwärtskomptabilität eingehalten werden.

## Soll-Kriterien

* Das bereits trainierte neuronale Netz soll nachträglich mit weiteren Daten trainierbar sein.
* Die erkannten Objekte sollen kategorisiert werden können. Hierzu soll eine eigene interne neue Anwendung entwickelt werden, welche ebenfalls ein vergleichbares Verfahren des maschinellen Lernens nutzt. Zur Abgrenzung soll dieses im Folgenden als *object identification application* bezeichnet werden.
* Die Anwendung erlaubt eine parallelisierte Verarbeitung mehrerer zu untersuchenden Bilddateien.
* Hierzu verfügt diese über eine Konfigurationsschnittstelle, um die maximale Zahl an zu verwendeten Prozessen *(Threads)* festzulegen.
* Die Anwendung soll IronPython kompatibel sein, um eine Ausführung auf einer Common Intermediate Language (bspw.: .NET) Laufzeitumgebung zu ermöglichen.

## Kann-Kriterien

* Die Anwendung soll in einen Web-Server eingebunden werden, welcher eine Schnittstelle zur Eingabe von Bilddaten, sowie eine Schnittstelle zur Ausgabe der analysierten Daten ermöglicht.
* Es wird eine iPhone-App zur Fotographie entwickelt, welche die erfassten Bilder an den oben genannten Web-Server übertragen kann. Weiterhin soll diese die von der Server-Anwendung erarbeiteten und ausgegebenen Daten empfangen und entsprechend darstellen können.
* Die Umsetzung des Trainings enthält eine Methode zur Kalkulation auf einer Grafikkarte, anstelle des Prozessors, um somit eine höhere Berechnungsgeschwindigkeit zu realisieren.
* Die Anwendung kann auch lokal auf einem iPhone (iOS 9) ausgeführt werden. Hierzu soll die Anwendung Cython kompatibel sein, um eine Portierung in einen Apple kompatiblen C Programmcode zu ermöglichen.
* Die Anwendung soll Jython kompatibel sein, um eine Ausführung Java Laufzeitumgebung zu ermöglichen.

## Abgrenzungskriterien

* Die Bilderkennungs-Anwendung enthält lediglich das neuronale Netz mit den trainierbaren Parametern, eine Ein- und Ausgabeschnittstelle zur Bildanalyse, sowie eine Möglichkeit eine Menge von gekennzeichneten Daten zum Training zu verwenden.
* Die Web-Server-Anwendung ermöglicht selbst keine Bilderkennung. Es wird lediglich eine Ausführung der oben beschriebene „object recognition & identification“ – Anwendung gewährleistet.
* Es gibt keine Einschränkung für den Benutzer, es gibt keine kostenpflichtigen Erweiterungen.
* Es können nur Objekte wiedererkannt werden, welche bereits durch ein umfangreiches Trainingsverfahren dem neuronalen Netz vorgestellt wurden. Um einen Vergleichsfaktor zur Einschätzung der Leistungsfähigkeit des neuronalen Netzes zu generieren, soll eine feste Menge an Objekten zur Trainings- und Testphasen festgelegt werden. Anhand dieser orientiert sich die Entwicklung des neuronalen Netzes. Für die Erkennung weiterer Objekte wird vom Endanwender die selbstständige Nutzung der Trainingsschnittstelle vorausgesetzt.
* Weiterhin wird eine den Trainingsdatensätzen ähnliche Qualität der Bilder zur Analyse vorausgesetzt.

# Produkteinsatz

## Anwendungsbereiche

Die Anwendung dient der automatisierten Verarbeitung von unstrukturierten Informationen in Form von Bildern, um eine hohe personelle Belastung zur iterativen Verarbeitung dieser zu vermeiden. Dies soll weiterhin die automatische Suche nach Begriffen von nicht näher bezeichneten Bildern ermöglichen. Denkbar wäre eine nachträgliche Indexierung der Ermittelten Objektinformationen um die Anwendung in eine Bilder-Datenbank zu integrieren.

## Zielgruppe

Die App ist sowohl für den privaten, als auch für den kommerziellen Gebrauch geeignet. Insbesondere Entwickler und Betreiber von Bilder-Datenbanken erhalten somit eine zeitsparende Möglichkeit, Bilder nicht ihren Informationen nach manuell zu beschreiben, sondern ein automatisiertes Verfahren zur Ermittlung der benötigten Informationen zu nutzen. Hierzu wird eine automatisierte textbasierte Suchbegriff-Indizierung ermöglicht.

## Betriebsbedingungen

* Der Betrieb der Anwendung setzt eine Common Intermediate Language RunTime voraus.
* Der Betrieb der Anwendung innerhalb des Servers setzt eine pythonfähige Laufzeitumgebung mit einer Internet-Verbindung voraus.
* Der Betrieb der App setzt eine funktionierende Kamera des iPhones voraus.

# Produktumgebung

## Software

Die Anwendung wird für eine Python Laufzeitumgebung der Version 2.7.10, oder innerhalb der zweiten Python Generation auch höher, entwickelt. Durch die Komptabilität zu IronPython und Jython wird ebenfalls eine Ausführung auf der Java bzw. Common Intermediate Language Laufzeitumgebung ermöglicht, nicht jedoch als primäres Ausführmedium für die Entwicklung festgelegt.

## Hardware

Für die Hardware wird die Ausführung eines zur Python Laufzeitumgebung kompatiblen Betriebssystems vorausgesetzt. Weiterhin wird eine Hardware-Umgebung vorausgesetzt, welche dem Umfang der Anwendung entsprechend genug physikalische und virtuelle Ressourcen zur Verfügung stellt.

# Produktfunktionen

|  |  |
| --- | --- |
| /F10/ | Import von Bildinformationen |

Über die universelle Eingabe-Schnittstelle des neuronalen Netzes (vgl. Schnittstellen) werden Bild-Informationen eingelesen und für die Funktionalität F20 vorverarbeitet bereitgestellt.

|  |  |
| --- | --- |
| /F20/ | Analysieren von Bildinformationen |

Die der Anwendung zur Verfügung gestellten Bildinformationen sollen durch einen internen Mechanismus gemäß den abgebildeten Objekten ausgewertet werden. Hierbei werden die auf dem Bild vorhandenen Objekt erkannt und in innerhalb der Form eines geometrischen Rechtecks erfasst. Die Rechtecke werden hierbei durch die Koordinaten auf dem Bild beschrieben.

|  |  |
| --- | --- |
| /F30/ | Export der Analyse |

Die analysierten Daten sollen über eine Ausgabe-Schnittstelle an den ursprünglich aufrufenden Prozess übertragen werden. Die Informationen der erkannten Objekte sollen hierbei gemäß der Schnittstellendefinition in eine Instanz einer Informationsklasse gekapselt werden. Diese enthält eine Liste von Rechtecken, definiert durch vier Koordinatenpunkte innerhalb des Bildes, welche die Position des erkannten Objektes angeben.

|  |  |
| --- | --- |
| /F40/ | Initialisierung der Kanten der *object recognition* |

Der Anwender erhält die Möglichkeit die Intelligenz, abgebildet durch die Kantengewichte, mit einem Standart-Wert zu initialisieren. Hierbei wird keine trainierte Version, sondern der tatsächlich untrainierter Initialwert „1“ auf alle Kantengewichte gesetzt.

|  |  |
| --- | --- |
| /F50/ | Trainieren des neuronalen Netzes |

Über eine Schnittstelle, welche jeweils ein Bild, sowie die hierzu korrekt analysierten Informationen enthält, soll das neuronale Netz trainiert werden können. Hierzu wird ein supervised-learning Verfahren angewendet.

|  |  |
| --- | --- |
| /F60/ | Export der Intelligenz |

Die in den Kantengewichten hinterlegte Intelligenz soll in eine namentlich freiwählbare Binär- oder XML-Datei exportierbar sein.

|  |  |
| --- | --- |
| /F70/ | Import der Intelligenz |

Über eine Schnittstelle werden die oben genannten Binär- oder XML-Dateien zur Initialisierung des neuronalen Netzes importiert.

# Qualitätsbestimmungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Anforderung | sehr wichtig | wichtig | neutral | irrelevant |
| Zuverlässigkeit |  |  | x |  |
| Stabilität |  | x |  |  |
| Performance |  |  |  | x |
| Benutzerfreundlichkeit |  |  |  | x |
| Oberfläche |  |  |  | x |
| Funktionalität | x |  |  |  |
| Portabilität |  | x |  |  |
| Wartbarkeit |  |  | x |  |

# Testszenarien und Testfälle

Es werden zwei Datenmengen von Bilder unterschieden. Die erste sei hierbei die Trainingsdatenmenge, welche alle Daten zum Trainieren einsetzt. Die zweite Datenmenge ist die Testdatenmenge, welche zur Überprüfung der Leistungsfähigkeit der Anwendung genutzt werden soll. Dieser Leistungstest ist insbesondere daher wichtig, da dieser den Faktor zur Bemessung und somit Vergleichsmerkmal innerhalb der Entwicklung abbildet. Dieser Faktor gilt als maßgeblicher Leitfaden für die Entwicklungsrichtung des neuronalen Netzes.

Hierfür soll die standardisierte *CIFAR-100 dataset* Datenmenge verwendet werden, welches sowie Trainingsdaten als auch Testdaten enthält. Durch den hohen Umfang dieser können zwar Leistungstest feingranular durchgeführt werden, jedoch erhöht eine hohe Menge an Daten ebenfalls die zeitliche Dauer des Trainings. In der frühen Entwicklungsphase soll vorerst ein grobgranularer Vergleich ausreichen. Daher soll in jener Phase vorerst die minimierte Version *CIFAR-10 dataset* genutzt werden, um eine effizientere Entwicklung zu ermöglichen.

Sollte eine höhere Varianz der Daten, oder ein höher Umfang an Daten benötigt werden, so wird auf die ebenfall standardisierte Datenmenge *NORB Object Recognition DataSet* (nachfolgend *NORB*), sowie deren minimierte Version *small NORB* zurückgegriffen werden.

s

Todo: beispielhaft ein paar Aussuchen (cifar10)

# 

# Schnittstellen

## Bildinformationen

### Import

Das neuronale Netz verfügt über eine universelle Eingabe-Schnittstelle über welche Bild-Informationen eingelesen werden können. Um die Anwendung kompatibel zu halten, ist neben den üblichen Bildformaten *Portable Network Graphics*, *JPEG*, *Tagged Image File Format* und *Bitmap* auch eine Reindatendarstellung in Form eines zweidimensionalen Daten-Arrays möglich.

### Export

Die analysierten Daten werden über eine Ausgabe-Schnittstelle an den ursprünglich aufrundenden Prozess übertragen. Hierbei werden die Informationen der Objekte in ein Schnittstellenobjekt gekapselt, welche die vier Koordinatenpunkte des Rechtecks, sowie optional die Informationen der zu erkennenden Objekte beinhält.

## Intelligenz

Für den universellen Datenaustausch, wird ein allgemeines XML-Format für den Austausch der Intelligenz des neuronalen Netzes entwickelt.

### Import

Es wird eine Schnittstelle implementiert, welches das oben genannte XML-Format importiert und das neuronale Netz mit der darin enthaltene Intelligenz initialisiert.

### Export

Es wird eine Schnittstelle implementiert, welche die trainierte Intelligenz(Kantengewichte) in das obengenannte XML-Format überführt und über einen Stream exportiert.

# Entwicklungsumgebung

## Software

Die App wird in der Programmiersprache *Python* entwickelt. Verwendete Softwares sind:

* OS X 10.11.1 (El Capitan)
* Linux Ubuntu Trusty Tahr (14.04 LTS)
* Nvidea CUDA
* Sublime Text
* Xcode 7.1.1
* Python PyPy
* IronPython
* Jython
* Cython
* Github Desktop zur git Versionskontrolle

## Hardware

Zum Testen stehen die folgenden Geräte zur Verfügung:

* iPhone 6s
* iPad mini (1. Generation)