HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

reformierte
kirche horgen

Bildklassifikation mit Hilfe eines Neuralen Netzes

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2018

Autoren: Martin Odermatt, Tobias Saladin
Betreuer: Prof. Oliver Augenstein
Projektpartner: Evangelisch-reformierte Kirche Horgen

Inhaltsverzeichnis

Abstract

Die Evangelisch-reformierte Kirche Horgen ist im Besitz einer unstrukturierten Bildersammlung, die eine Suche nach Bildern mit bestimmen Merkmalen nicht unterstützt. Die Bilder unterliegen dem Datenschutz, wodurch eine Klassifizierung durch externe Cloud-Dienste nicht gestattet ist. Das Ziel der vorliegenden Studienarbeit war es, eine Machbarkeitsanalyse mit dem Inception-v3 Convolutional Neural Network durchzuführen. Dabei sollte gezeigt werden, ob sich das Netzwerk auch für die Klassifizierung der Bilder der Evangelisch-reformierten Kirche Horgen eignet.

Da die vom Inception-v3 Netzwerk vorgegebenen Kategorien nicht zu den Bildern der Evangelisch-reformierten Kirche Horgen passten, wurde der letzte Layer des Netzwerkes abgeschnitten. Aus den verbleibenden Layern wurde ein siamesisches Netzwerk aufgebaut, mit dem Bildpaare auf ihre Ähnlichkeit hin verglichen werden konnten. Um die Güte des siamesischen Netzwerkes zu messen, wurde mit einer bereits kategorisierten, frei verfügbaren Bilddatenbank, die nicht für das Training des Netzwerkes verwendet wurde, eine ROC-Kurve aufgezeichnet. Anhand dieser Kurve wurden Hyper-Parameter, wie der Threshold für das Clustering und ein Ähnlichkeitsmass, festgelegt. Die Methode der Messung war an dieser Stelle die ROC-Kurve. Die Ergebnisse wurden anschliessend anhand der Kurve analysiert. Die Kosinus-Ähnlichkeit erzielte in Kombination mit den Feature-Vektoren den höchsten AUC-Wert von 0.972 und wurde somit für den Anwendungsfall als geeignet beurteilt. Mit Hilfe des dazugehörigen Threshold-Werts wurde ein Clustering-Algorithmus entwickelt, der durch die Bestimmung möglichst unterschiedlicher Bilder einen Überblick über die Bildersammlung geben soll. Um auch eine auf Schlüsselwörtern basierende Bildersuche zu unterstützen, wurde die Anwendung mit der Google-Bildersuche verknüpft, sodass von Google gelieferte Bilder als Quelle für die Suche ähnlicher Bilder verwendet werden können.

Im Zuge der Studienarbeit ist ein Prototyp entstanden, der nach dem einmaligen Import einer Bildersammlung durch das Inception-v3 Netzwerk automatisch Feature-Vektoren berechnet und die Bilder in Cluster aufteilt. Nach dem Import können Bilder aus den Clustern oder aus dem Resultat der integrierten Google-Bildersuche als Referenz für die Suche nach ähnlichen Bildern ausgewählt werden. Als Resultat werden dann die Ergebnisse mit der höchsten Ähnlichkeit in der Anwendung angezeigt.

Der erstellte Prototyp lieferte für den Anwendungsfall bereits ohne zusätzliches Training zufriedenstellende Ergebnisse. Limitationen waren vor allem bei Suchanfragen mit mehreren oder abstrakten Merkmalen sowie beim Erkennen von spezifischen Personen gegeben.

Management Summary

Ausgangslage

Die Evangelisch-reformierte Kirche Horgen ist im Besitz einer unstrukturierten Bildersammlung. Eine gezielte Suche nach Bildern mit bestimmten Merkmalen wird daher nicht unterstützt. Auch unterliegen die Bilder dem Datenschutz, was eine Aufteilung in verschiedene Kategorien durch externe Cloud-Dienste nicht gestattet.

Das Ziel der vorliegenden Studienarbeit war es, eine Machbarkeitsanalyse durchzuführen. Diese sollte zeigen, ob eine Durchsuchung der Bildersammlung der Evangelisch-reformierten Kirche Horgen mit Schlüsselworten oder anhand ähnlicher Bilder möglich ist, ohne dabei gegen den Datenschutz zu verstossen.

Vorgehen und Technologien

Um eine Kategorisierung der Bilder zu ermöglichen, wurde eine Architektur, basierend auf dem von Google frei zur Verfügung gestellten Neuralen Netzwerk, verwendet. Das Neurale Netzwerk ist in der Lage, ein Bild aus verschiedenen Kategorien zu erkennen. Das Netzwerk wurde für den Anwendungsfall der Evangelisch-reformierten Kirche Horgen modifiziert, da die vom Neuralen Netzwerk vorgegebenen Kategorien nicht zu den Bildern passten. Anschliessend wurde die Architektur in verschiedenen Varianten getestet. Für den zu entwickelnden Prototypen konnte so die geeignetste Variante ausgewählt und integriert werden.

Um auch eine auf Schlüsselwörtern basierende Bildersuche zu unterstützen, wurde die Anwendung mit der Google-Bildersuche verknüpft. So können von Google gelieferte Bilder als Quelle für die Suche ähnlicher Bilder verwendet werden. Weiter sollte die Anwendung nach dem Start eine Übersicht über die Bildersammlung anzeigen. Dazu wurde ein Verfahren entwickelt, mit dem sich die Bilder nach Ähnlichkeiten gruppieren lassen.

Ergebnisse

Im Zuge der Studienarbeit zeigte sich, dass die erstellte Architektur für den Anwendungszweck zufriedenstellende Ergebnisse lieferte. Der Prototyp ermöglicht dank der verknüpften Google-Bildersuche eine Durchsuchung der Bildersammlung anhand von Schlüsselwörtern. Ebenfalls wird das Auffinden von ähnlichen Bildern zu einem ausgewählten Referenzbild unterstützt. Zudem wird beim Start der Anwendung, dank der Gruppierung, eine Übersicht über die Bilder angezeigt.

Limitationen waren vor allem bei Suchanfragen mit mehreren oder abstrakten Schlüsselwörtern sowie beim Erkennen von spezifischen Personen gegeben. Der entstandene Prototyp wurde bei einer Präsentation von den Mitarbeitenden der Evangelisch-reformierten Kirche Horgen sehr positiv aufgenommen.

Ausblick

Da die Machtbarkeitsanalyse vielversprechend ausfiel, wurde entschieden, das Projekt in einer Bachelorarbeit weiter zu verfolgen und zu vertiefen. Dabei sollen Optimierungen im Bereich der Benutzerfreundlichkeit und der Performanz im Fokus stehen sowie auf Wünsche seitens der Evangelisch-reformierte Kirche Horgen eingegangen werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Technischer Bericht

1 Ausgangslage

Die vorliegende Studienarbeit beschreibt eine Machbarkeitsanalyse. Geprüft wurde, ob es möglich ist, mit Hilfe eines Convolutional Neural Network eine Applikation zu entwickeln, welche es der benutzenden Person erlaubt, ihre persönliche Bildersammlung nach ähnlichen Bildern zu durchursuchen. Aufgrund der Ergebnisse der Analyse wurde anschliessend entschieden, ob und wie eine Applikation für den beschriebenen Anwendungsfall erstellt werden soll.

Bezüglich der durch die Begleitperson definierten Aufgabenstellung wird auf Anhang A verwiesen.

1.1 Heutige Praktiken

Die Evangelisch-reformierte Kirche Horgen verfügt über eine Bildersammlung von 15’295 Bildern. Diese Bilder sind unsortiert auf einer Festplatte gespeichert. Wird nach einem Bild mit bestimmten Charakteristiken gesucht, muss die komplette Ordnerstruktur durchforscht werden. Auch liefert eine Bildersuche mittels eines Suchbegriffs häufig keine Übereinstimmungen, da die Ordner und Bilder nicht treffend benannt wurden. Viele Bilder, welche zum Beispiel für einen Flyer in Frage kommen würden, werden dabei übersehen und gelangen nie in die engere Auswahl.

1.2 Zielgruppe

Die Machbarkeitsanalyse dieser Studienarbeit richtet sich an Fachexpertinnen und Fachexperten auf dem Gebiet der Bilderkennung mit Hilfe von Neuralen Netzwerken.
Das Endprodukt, ein Prototyp für das Suchen ähnlicher Bilder, richtet sich an die Mitarbeitenden der Evangelisch-reformierten Kirche Horgen.

1.3 Vorgegebene Technologien

Für diese Studienarbeit wurden einige Technologien durch die Begleitperson vorgegeben. Dies waren die Programmiersprache Python [2], das für maschinelles Lernen und künstliche Intelligenz entwickelte Framework TensorFlow [3] sowie das Convolutional Neural Network (CNN) Inception-v3. Nachfolgend wird das Inception-v3 Netzwerk beschrieben.

TensorFlow

TensorFlow ist eine leistungsstarke Open-Source-Softwarebibliothek für numerische Berechnungen. Ursprünglich wurde TensorFlow von einem Team aus Forschern und Ingenieuren bei Google Brain entwickelt [4]. Die plattformübergreifende Programmierschnittstelle erlaubt es Algorithmen des Machine Learnings auszuführen und bietet komfortable Wrapper-Funktionen für Deep Learning. Google selbst setzt TensorFlow in einigen Diensten produktiv ein und es hat sich in der maschinellen Lernforschung durchgesetzt [5].

Inception-v3 [1]

Inception-v3 wurde von Google für die Klassifizierung von Bilddateien entwickelt und kategorisiert diese in insgesamt 1000 vorgegebene Klassen. In dieser Studienarbeit wird die Version 3 des Netzwerkes verwendet. Es ist eine Weiterentwicklung des AlexNet und scheint für die Studienarbeit geeignet, da mit verhältnismässig geringem Rechenaufwand eine sehr hohe Klassifizierungsgenauigkeit erreicht werden kann. Inception-v3 erreichte bei der ImageNet Challenge 2012 eine Top-5 Fehlerquote von 3.46%.

Bilder bestehen aus vielen einzelnen Pixeln, daher ist die Verarbeitung selbst mit modernen Maschinen sehr aufwändig. Es entsteht eine grosse Anzahl an Inputs und verschiedenen Layern. Um diesem Umstand entgegen zu wirken wird ein Convolutional Neural Network, wie zum Beispiel Inception-v3, eingesetzt.

Beim CNN wird die Aktivität der Neuronen über eine diskrete Faltung berechnet. Diese bündelt die einzelnen Pixel zu sogenannten Merkmalen. Ein farbiges Bild kann beispielsweise in Höhe und Breite sowie in drei Spektralfarben aufgeteilt werden. Nun wird eine Faltungsmatrix, wie in Abbildung 1 ersichtlich, über die einzelnen Bildpunkte bewegt und daraus die Gewichtung berechnet. Diese wird an das nächste Layer übergeben.

Die Grösse der Faltungsmatrix bestimmt, wie viele Pixelwerte miteinander addiert werden.
Dies soll anhand eines Beispiels veranschaulicht werden: Die Faltungsmatrix hat eine Höhe und Breite von je fünf Teilen und eine Tiefe von drei Teilen. Daraus entsteht ein Würfel von insgesamt 75 Teilen. Nun werden diese 75 Teile jeweils mit dem Gewicht multipliziert und addiert. Das entstandene Resultat wird an die nächste Schicht weitergegeben. Durch die Faltungsmatrix wird erreicht, dass benachbarte Neuronen einen gewissen Einfluss aufeinander haben. Somit lassen sich einzelne Merkmale immer wieder bündeln, um so schlussendlich Klassen zu definieren. Bei diesem Vorgang wird die Anzahl Neuronen kleiner, die Anzahl Klassen nimmt jedoch zu.

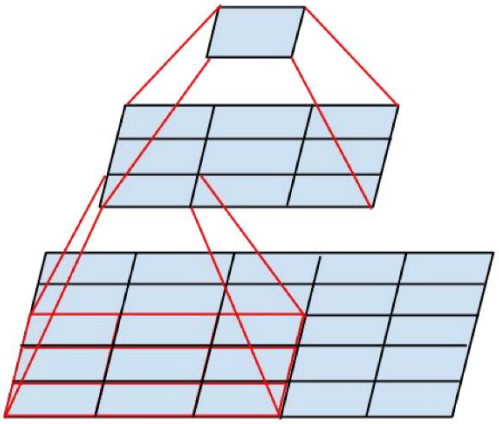

Abbildung 1: Prinzip der Faltungsmatrix

Anmerkung. Die Abbildung bezieht sich auf die Quelle [1].

2 Problembeschreibung

Dieses Kapitel geht auf Problemstellungen, Anforderungen sowie Risiken ein, welche im Zusammenhang mit dieser Studienarbeit berücksichtigt werden müssen.

2.1 Analyse von bereits bestehenden Applikationen

Im Internet lassen sich viele Programme finden, die in erster Linie dazu gedacht sind, Bilder zu bearbeiten. Nur wenige bieten eine Funktionalität für Bildvergleiche an. In Tabelle 1 sind die gängigsten im Internet auffindbaren Programme aufgelistet.

Tabelle 1: Analyse der bereits bestehenden Applikationen

Produkt und Version	lokal / online	Feature ähnliche Bilder suchen	Letztes Update	Bemerkungen
XnView MP 0.92 [6]	lokal	nein	19.09.2018	Eine Funktion “Duplikate finden” ist vorhanden. Eine Suche lässt sich über beliebige Ordner starten. Es kann aber kein Referenzbild angegeben werden, d.h. die Suche resultiert mit allgemein gefundenen Duplikaten.
Google Cloud Vision API v1 [7]	online	nein	28.09.2018	Google bietet eine Programmierschnittstelle an, wodurch an Google gesendete Bilder automatisch mit einer Wahrscheinlichkeit gelabelt werden.
Awesome Duplicate Photo Finder 1.1 [8]	lokal	eingeschränkt	19.10.2012	Zu einem Bild kann jeweils ein ähnliches Bild gefunden werden. Dabei wird mittels einer Wahrscheinlichkeit angezeigt, wie ähnlich diese sind.
Imgseek 0.8.6 [9]	lokal	ja	17.03.2006	Bilder lassen sich aufgrund ihrer Ähnlichkeit gruppieren. Die Suche liefert laut Entwickler inhaltsähnliche Resultate. Es ist unter anderem auch möglich, ein Bild zu skizzieren und dieses als Ausgangslage zu verwenden.
VisiPics 1.31 [10]	lokal	ja	4.02.2013	Die Applikation bietet eine Suche nach ähnlichen Bildern an. Der Filterregler wird dazu auf “Loose” definiert, womit die Übereinstimmungsquote gesenkt wird.
Similar.Pictures [11]	lokal und online	eingeschränkt	19.11.2018	Ähnliche Bilder werden zu Clustern gruppiert. Zu jedem Bild wird ein Hash über einen RGB-Wert berechnet, danach werden mehrere Bilder mit dem Cosine-Algorithmus auf Ähnlichkeit untersucht. Es werden farblich und objektgleiche Bilder aus der gleichen Perspektive erkannt, d.h. echte Duplikate.
Google Photos [12]	online	ja	unbekannt	Damit der Dienst genutzt werden kann, wird ein Google-Account vorausgesetzt. Bilder werden in die Cloud hochgeladen.

Anmerkung. Eigene Darstellung.

2.1.1 Auswertung

Bis zum heutigen Zeitpunkt (Dezember 2018) lässt sich eine begrenzte Anzahl an Applikationen finden, die in der Lage sind, ähnliche Bilder zu identifizieren. Diese fokussieren insbesondere die exakten Farbwerte und liefern daher lediglich für “echte Duplikate” ausreichende Resultate. Auffallend ist ausserdem, dass die Hälfte der gefundenen Applikationen in den letzten fünf Jahren nicht mehr aktualisiert wurden. Grundsätzlich lassen sich die gefundenen Applikationen in vier Kategorien einteilen, wobei “Google Photos” allen Kategorien zugeordnet werden kann. Diese Kategorien sind in Tabelle 2 ersichtlich.

Tabelle 2: Kategorien von Bildvergleichs-Programmen

	Kategorie 1	**Kategorie 2**	**Kategorie 3**	**Kategorie 4**
Typ	Auffinden von echten Duplikaten anhand von Metadaten	Auffinden von ähnlichen Bildern über RGB-Werte	Bilder lassen sich manuell oder automatisch mit Hilfe von Schlagwörtern kategorisieren	Gesichtserkennungs-Software
Applikation	• XnView MP • Awesome Duplicate Photo Finder	• Imgseek • VisiPics • Similar.Pictures	• Google Cloud Vision API	
	Google Photos			

Anmerkung. Eigene Darstellung.

2.1.2 Schranken

In den Kategorien 3 und 4 gibt es mittlerweile Lösungen von Apple, Microsoft, Google und anderen Anbietern. Diese wurden jedoch ausser Acht gelassen, da der Fokus dieser Studienarbeit nicht in diesen Kategorien liegt.

2.1.3 Fazit der Analyse

Einzig die Applikation “Google Photos” ist in der Lage, ähnliche Bilder anzuzeigen. Alle anderen suchen nach identischen Bildern anhand von Metadaten oder vergleichen Bilder anhand der RGB-Werte und sind daher nicht geeignet. ImgSeek verspricht auch ähnliche Bilder zu finden. In einem Praxistest hat sich herausgestellt, dass die Trefferquote sehr gering ausfällt. Beispielsweise wurde bei einer Suche nach einer Kirchenorgel ein Bild einer Ameise als bester Treffer angegeben. Weitere Kirchenorgeln wurden nicht gefunden. Ein Cloudanbieter wie Google kommt aus datenschutzrechtlichen Gründen für die Evangelisch-reformierte Kirche Horgen nicht in Frage. Letztendlich bleibt das Entwickeln einer eigenständigen Software zur Kategorisierung der Bilder als einzige Option bestehen. Allenfalls könnten die Bilder auch von einem externen Anbieter mit unter Berücksichtigung des Datenschutzes klassifiziert werden.

2.2 Anforderungsspezifikation

Die Anforderungen werden nachfolgend in funktionale und nicht-funktionale Anforderungen unterschieden.

2.2.1 Funktionale Anforderungen

In Abbildung 2 sind neun funktionale Anforderungen in Form von Use Cases dargestellt, die in Tabelle 3 erläutert werden.

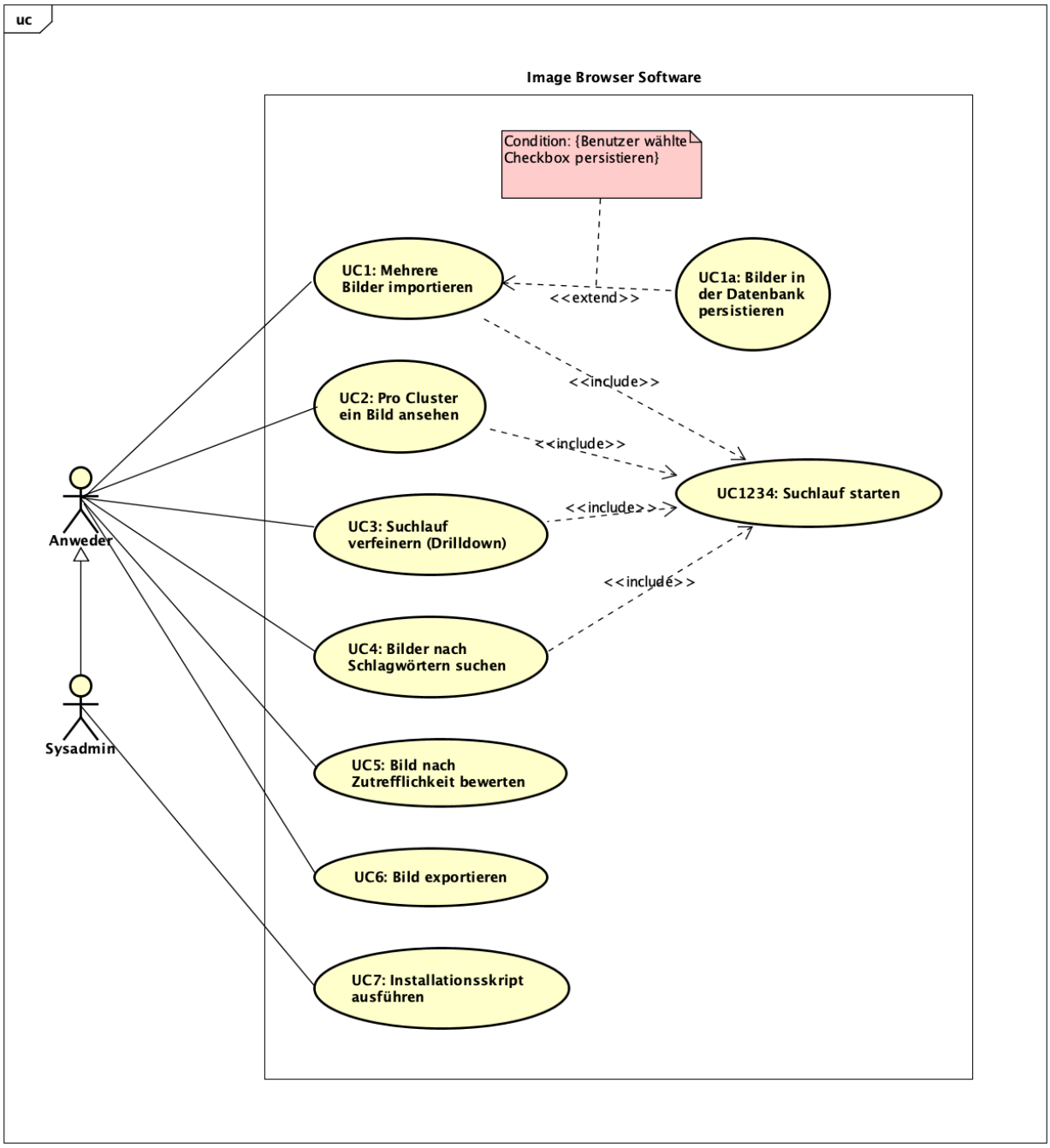

Abbildung 2: Funktionale Anforderungen als Use Cases

Anmerkung. Eigene Darstellung.

Tabelle 3: Funktionale Anforderungen

UC	Titel	Beschreibung
UC1	Mehrere Bilder importieren	Ein Benutzer kann mehrere Bilder auf einmal importieren.
UC1a	Bilder in der Datenbank persistieren	Ein Bild kann entweder permanent in die Datenbank aufgenommen oder lediglich für eine Suche verwendet werden.
UC2	Pro Cluster ein Bild ansehen	Ein Benutzer kann eine zufällige Suche starten. Dabei wird von jedem Cluster mindestens ein Bild angezeigt.
UC3	Suchlauf verfeinern (Drilldown)	Eine Suche wird aufgrund der vom Benutzer ausgewählten Bilder verfeinert.
UC4	Bilder nach Schlagwörtern suchen	Es kann nach einem Bild aufgrund von eingegebenen Schlagwörtern gesucht werden.
UC5	Bild nach Zutrefflichkeit bewerten	Für das Training ist es wichtig, dass ein Bild als passend markiert werden kann.
UC6	Bild exportieren	Gefundene Bilder können direkt auf das Filesystem exportiert werden.
UC7	Installationsskript ausführen	Die Installation soll soweit möglich von einem Systemadministrator über ein Skript erfolgen.
UC1234	Suchlauf starten	Ein Suchlauf wird jeweils bei UC1, UC2, UC3 und UC4 automatisch oder nach einem manuellen Anstoss gestartet.

Anmerkung. Eigene Darstellung.

2.2.2 Nicht-funktionale Anforderungen

Da die Applikation prototypartig entwickelt wird, lassen sich nur wenige nicht-funktionale Anforderungen ableiten, die in Tabelle 4 festgehalten sind. Für die Klassifikation wird das ”FURPS+“ Akronym verwendet, das ursprünglich bei Hewlett-Packard entwickelt wurde und heute vielerorts in der Softwareindustrie anzutreffen ist [13].

Tabelle 4: Nicht-funktionale Anforderungen

Nr.	FURPS+	Titel	Beschreibung
1	Performance	Feature-Berechnung	Die Berechnung der Features pro Bild soll mit der vorhandenen Hardware innert eines Arbeitstages abgeschlossen sein.
2	Supportability	Win 10	Die Applikation soll auf dem Windows 10 Betriebssystem laufen.
3	+	Datenschutz	Aus datenschutzrechtlichen Gründen dürfen die Bilder aus der Bilddatenbank nicht an Dritte weitergegeben werden und sind nur lokal zu speichern.
4	Usability	Zusätzliches Training	Die Benutzerführung für die Bewertung von Bildern, die dem zusätzlichen Training dient, soll intuitiv sein, d.h. pro Bild soll es zwei Optionen in Form von ”zutreffend“ und ”nicht zutreffend“, geben.

Anmerkung. Eigene Darstellung.

2.3 Risikomanagement

In Tabelle 5 wurden relevante Risiken zusammengetragen und ausgewertet. Um negative Auswirkungen auf den Projektverlauf zu verhindern, wurden die beschriebenen Massnahmen definiert und teilweise bereits umgesetzt.

Tabelle 5: Risikomanagement

Nr.	Kategorie	Beschreibung	Folgen	Eintrittswahrscheinlichkeit	Auswirkung	Vorbeugung	Verhalten bei Eintreten
R1	Security	Datenschutz wird nicht eingehalten	Rechtliche Konsequenzen	unwahrscheinlich	KO	Daten werden ausschliesslich lokal bearbeitet	Weiteres Vorgehen mit dem Dozenten besprechen
R2	Performance	Verarbeitung der Bilddaten dauert zu lange und beeinflusst die User Experience negativ	Applikation wird nicht mehr verwendet	sehr gross	kritisch	Daten werden durch GPU verarbeitet, Vergleich verschiedener Verarbeitungsmöglichkeiten	Profiling, Performance-Tuning
R3	Usability	Vorgeschlagene Bilder entsprechen nicht den Erwartungen des Benutzers	Applikation wird nicht verwendet	gross	kritisch	Netz manuell trainieren	Bildverarbeitung neu evaluieren
R4	Scope	Scope kann in vorgegebener Zeit nicht erfüllt werden	Produkt kann nur eingeschränkt fertig gestellt werden	gross	gering	Regelmässige Meetings mit dem Dozenten, laufende Beurteilung des Fortschritts	Scope mit Dozenten neu festlegen

Anmerkung. Eigene Darstellung.

Anschliessend werden die erfassten Risiken in einer Risikomatrix zusammengetragen.

Tabelle 6: Risikomatrix

		Auswirkung			
Eintrittswahrscheinlichkeit		Gering (1)	Mässig (2)	Kritisch (3)	KO (4)
	Sicher (5)				
	Sehr gross (4)			R2	
	Gross (3)	R4		R3	
	Mässig (2)				
	Unwahrscheinlich (1)				R1

Anmerkung. Eigene Darstellung.

Wie die Risikomatrix in Tabelle 6 zeigt, muss besonders Nummer R2 (Performance - Verarbeitung der Bilddaten dauert zu lange und beeinflusst die User Experience negativ) berücksichtigt werden.

2.4 Infrastruktur & Tools

Bezüglich der verwendeten Infrastruktur und Tools wird auf den Anhang E verwiesen.

3 Lösungskonzept

In diesem Kapitel werden die Lösungsansätze für die Umsetzung der zuvor definierten Anforderungen aufgezeigt. Im Zentrum standen die Qualitätsprüfung der Bildvergleiche sowie grundlegende Architekturentscheide der Anwendung.

Um die Qualität nicht nur visuell anhand unklassifizierter Bilder beurteilen zu müssen, wurde mit einer bereits klassifizierten Bildersammlung überwachtes Lernen [14] durchgeführt.

3.1 Tests mit Supervised Learning

Als Testumgebung wurde ein Programm erstellt, welches die Bildvergleiche mit Supervised Learning binär klassifiziert. Für die Entscheidungsfindung bedienten sich die Autoren dem Konzept der siamesischen Netzwerke. Das Programm plottet die Ergebnisse aus der siamesischen Architektur mit verschiedenen Schwellwerten in einer Receiver Operating Characteristic (ROC) Kurve [14].

3.1.1 Siamesische Netzwerke

Siamesische Netzwerke in ihrer ursprünglichen Form, werden im Bereich des Trainings vor allem für die Gesichtserkennung oder für Vergleiche von Teilen von Bildern eingesetzt. Iaroslav Melekhov, Juho Kannala und Esa Rahtu beschreiben in ihrem Bericht jedoch eine siamesische Architektur, welche für den Vergleich ganzer Bilder genutzt werden kann [15]. Anstatt verschiedene Bilder unterschiedlichen Kategorien zuzuordnen, wird mit der im Bericht beschriebenen siamesischen Architektur binär klassifiziert, ob sich zwei Bilder ähnlich sind oder nicht. Aufgrund dieser Erkenntnisse wählten die Autoren einen ähnlichen Ansatz.

3.1.2 Testaufbau

Für den Aufbau des siamesischen Netzwerkes benötigten die Autoren klassifizierte Bilder. Da die Bilder der Evangelisch-reformierten Kirche Horgen keine Labels enthalten, wurde die bereits klassifizierte Bildersammlung Caltech101 [16] verwendet. Diese enthält 101 Kategorien mit 40 bis 800 Bildern pro Kategorie.

Die für diese Studienarbeit verwendete siamesische Architektur besteht aus zwei parallelen Inception-v3 Netzwerken, welche komplett identisch sind. Die Netzwerke werden anschliessend zu einem Loss-Layer zusammengefasst. Im Loss-Layer wird die Distanz zwischen den Ausgaben der beiden Inception-v3 Netzwerke berechnet. Eine vereinfachte Darstellung der Architektur ist in Abbildung 3 ersichtlich. Der Vorgang kann folgendermassen beschrieben werden:

- Jedem Inception-v3 Netzwerk wird ein bereits klassifiziertes Bild zur Bearbeitung gegeben, zum Beispiel Bild Auto und Bild Blume
- Verarbeitung von Bild Auto durch das Inception-v3 Netzwerk 1
- Verarbeitung von Bild Blume durch das Inception-v3 Netzwerk 2
- Im Loss-Layer wird aus den erhaltenen Vektoren der Inception-v3 Netzwerke die Distanz berechnet
- Anhand eines Schwellwertes wird anschliessend entschieden, ob sich die Bilder ähnlich sind oder nicht

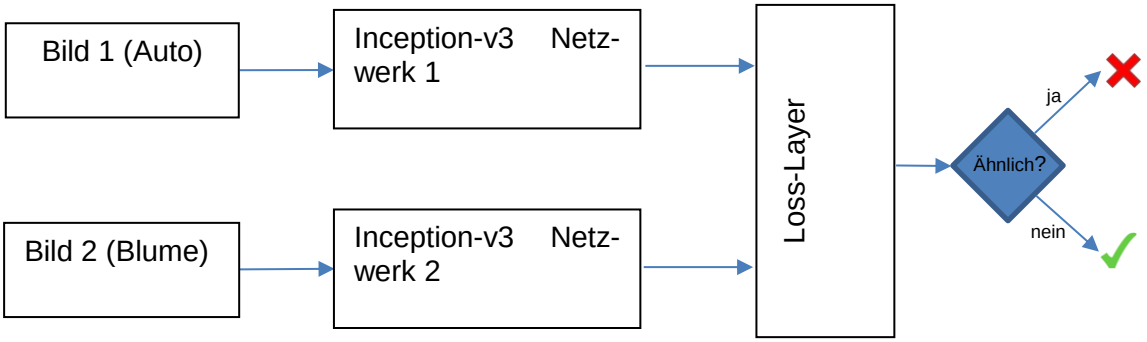

Abbildung 3: Vereinfachter Aufbau des Siamesischen Netzwerkes

Anmerkung. Eigene Darstellung.

Für die Berechnung der Distanzwerte wurde Manhattan-Metrik [17] verwendet. Um zu evaluieren, ob sich diese Metrik eignet, wurde als Vergleich die Kosinus-Ähnlichkeit [18] in die Testfälle miteinbezogen.
Das Inception-v3 Modell wurde für die Testfälle grösstenteils in der Ursprungsform belassen. Lediglich das Abschneiden des letzten Layers (der Softmax-Aktivierungsfunktion), wurde als mögliche Optimierung getestet. Als Konsequenz lieferte das Modul statt Wahrscheinlichkeiten aus 1000 Klassen 2048-dimensionale Feature-Vektoren.

Folglich ergaben sich insgesamt vier verschiedene Testszenarien:

1. Feature-Vektoren in Kombination mit Kosinus-Ähnlichkeit
2. Feature-Vektoren in Kombination mit Manhattan-Metrik
3. Wahrscheinlichkeiten in Kombination mit Kosinus-Ähnlichkeit
4. Wahrscheinlichkeiten in Kombination mit Manhattan-Metrik

3.1.3 Distanzberechnungen mit Manhattan und Kosinus

Für die Berechnung mit der Kosinus-Ähnlichkeit und der Manhattan-Metrik müssen vorerst die Ausgaben des Inception-v3 Modells genau analysiert werden:

Feature-Vektoren:
Die einzelnen Dimensionen eines Feature-Vektors sind ausschliesslich positive Zahlen. Die Feature-Vektoren sind nicht normalisiert, weshalb die Summennorm auch grösser als 1 sein kann.

Klassifizierte Vektoren:
Wie bei den Feature-Vektoren sind auch die einzelnen Wahrscheinlichkeiten eines klassifizierten Vektors ausschliesslich positive Zahlen. Die klassifizierten Vektoren sind normalisiert und daher ergibt die Summennorm immer 1.

Für die ROC-Kurve müssen die Vektoren normalisiert sein. Aufgrund der Tatsache, dass sowohl die Dimensionen als auch die Wahrscheinlichkeiten stets positiv sind, ergibt die Berechnung mit der klassischen Kosinus-Ähnlichkeits-Formel immer ein Resultat zwischen 0 und 1. Um die Distanz zwischen zwei Vektoren mit der Kosinus-Ähnlichkeit zu erhalten wurde das Ergebnis von 1 subtrahiert, wie aus der nachfolgenden Formel zu entnehmen ist.

$$\cos(\Theta) = 1 - \frac{a * b}{\|a\|_2 \|b\|_2}$$

Um auch bei der Manhattan-Metrik ein Resultat zwischen 0 und 1 zu erhalten, mussten die Feature-Vektoren normalisiert werden. Daraus ergab sich die nachfolgende Formel.

$$d(a,b) = \sum_{i=1}^{2048} \left(\left| \frac{a_i}{\sum_{j=1}^{2048} a_j} - \frac{b_i}{\sum_{j=1}^{2048} b_j} \right| \right)$$

Die Algorithmen wurden in dieser Form in der siamesischen Architektur verwendet.

3.1.4 Visualisierung mit ROC-Kurve

Eine ROC-Kurve gab den Autoren die Möglichkeit die Ergebnisse mit verschiedenen Schwellwerten zu berechnen. Für jedes Testszenario wurde mit der „roc_curve“ Methode von sklearn.metrics [19] eine ROC-Kurve geplottet. Pro Kurve wurden rund 385’000 Distanzen berechnet.

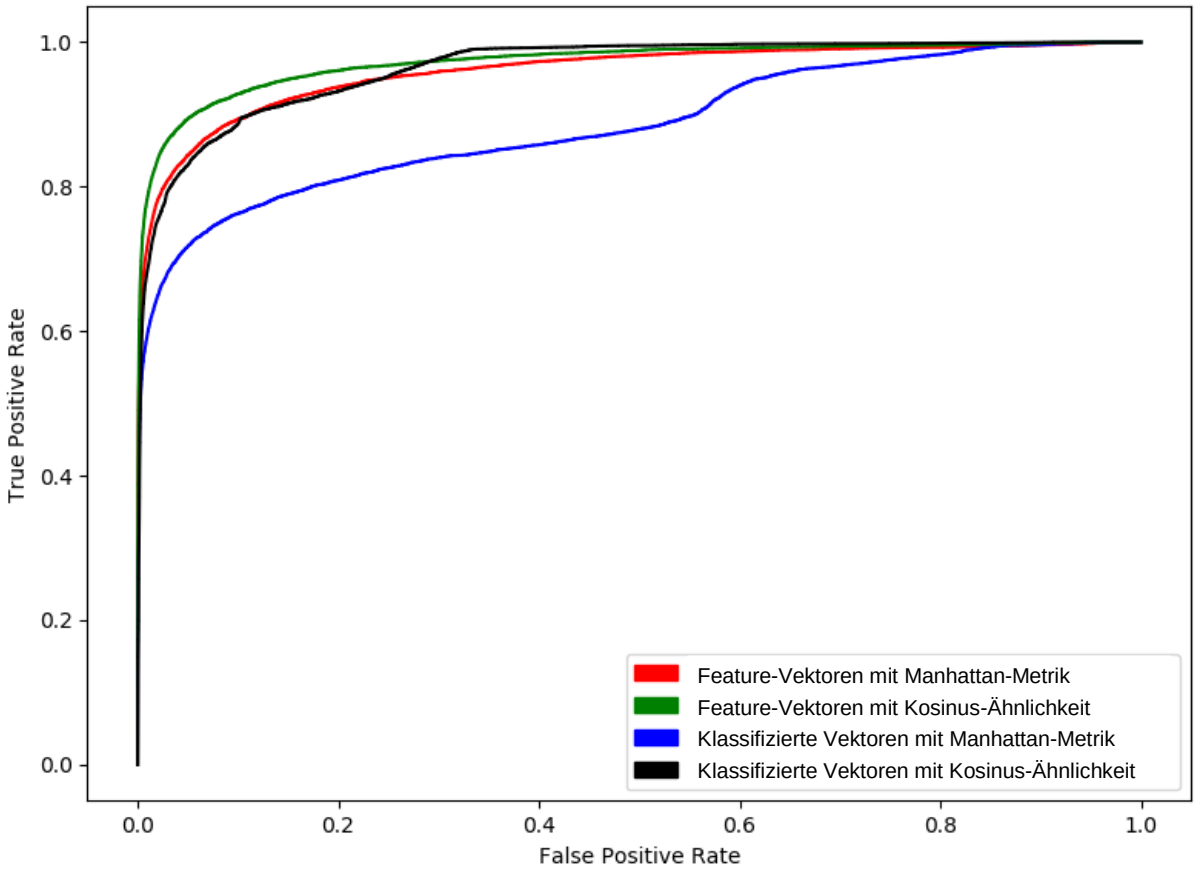

Abbildung 4: ROC Kurve

Anmerkung. Eigene Darstellung.

In Abbildung 4 ist ersichtlich, dass die grüne und die rote Kurve am linken oberen Rand mehr ausschwingen, als die schwarze und die blaue Kurve. Folglich liefert die Berechnung mit einem

hohen Schwellwert bei den Feature-Vektoren eine tiefere False Positive Rate als mit den klassifizierten Vektoren. Aus den vier Kurven ergaben sich zudem folgende Area under the curve (AUC) Werte:

- Feature-Vektoren mit Manhattan Metrik: 0.958
- Feature-Vektoren mit Kosinus Ähnlichkeit: 0.972
- Klassifizierte Vektoren mit Manhattan Metrik: 0.884
- Klassifizierte Vektoren mit Kosinus Ähnlichkeit: 0.963

Die Feature-Vektoren in Kombination mit der Kosinus-Ähnlichkeit erreichte dabei das beste Ergebnis. Auffällig ist, dass die Kombination der klassifizierten Vektoren mit der Manhattan-Metrik mit dem AUC-Wert von 0.884 leicht abfällt. Dies ist auch aus Abbildung 4 zu erkennen.

3.1.5 Performanz von Manhattan-Metrik und Kosinus-Ähnlichkeit

Für das Erstellen der ROC-Kurve wurden für die Algorithmen keine Libraries verwendet. Dies gab den Autoren die Möglichkeit die Formeln frei anzupassen. Um für die fertige Applikation die performanteste Lösung zu erhalten, wurden von der SciPy Bibliothek die spatial.distance.cityblock [20] Methode und die spatial.distance.cosine [21] Methode, auf ihre Performanz getestet. Die Ergebnisse sind in Tabelle 7 ersichtlich und bringen zum Ausdruck, dass sich der Einsatz der Distanzmethoden von SciPy lohnt. Dadurch kann die Geschwindigkeit der Berechnungen verdreifacht werden. Weiter ist aus der Tabelle zu entnehmen, dass die Manhattan-Metrik schneller rechnet als die Kosinus-Ähnlichkeit.

Tabelle 7: Zeitmessungen der Distanzberechnungen in Sekunden

Anzahl Berechnungen	**1'000**	**10'000**	**100'000**
SciPy Cityblock	0.09	1.00	9.91
SciPy Cosine	0.12	1.19	12.35
Manhattan ausprogrammiert	0.31	3.38	31.42
Kosinus ausprogrammiert	0.49	5.15	41.16

Anmerkung. Eigene Darstellung.

3.1.6 Fluch der Dimensionalität

Die Rechenzeiten der SciPy-Algorithmen wirken auf den ersten Blick ausreichend performant. Trotzdem sind die Zeitverzögerungen bei mehreren tausend Bildern für den Benutzer immer noch spürbar. Dies liegt vor allem am Fluch der Dimensionalität [14]. Dieser Begriff wurde durch Richard Bellmann definiert und beschreibt den Umstand, dass bei Zunahme der Anzahl Dimensionen die Problemlösung exponentiell ansteigt. Diverse Verfahren versuchen dem Fluch der Dimensionalität entgegenzuwirken. In dieser Studienarbeit wurde das Verfahren Principle Component Analysis in die Testfälle miteinbezogen.

3.1.7 Principle Component Analysis

Mit Hilfe einer Principle Component Analysis, ein unüberwachtes lineares Transformationsverfahren, wurde versucht, die Anzahl Dimensionen zu reduzieren [14]. Dabei stellt die PCA mit minimalem Aufwand eine Roadmap über komplexe Datensätze zur Verfügung und bringt die manchmal versteckte, einfache Struktur hervor [22].

Der PCA-Algorithmus besteht grundsätzlich ausfolgenden Schritten [4]:

1. Standardisierung der *d*-dimensionalen Datenmenge
2. Konstruieren der Kovarianzmatrix
3. Zerlegung der Kovarianzmatrix in Eigenvektoren und Eigenwerte
4. Sortieren der Eigenwerte in absteigender Reihenfolge, um eine Rangliste der Eigenvektoren zu erhalten
5. Auswahl der *k* Eigenvektoren, die zu den *k* grössten Eigenwerten gehören, wobei *k* die Dimensionalität des neuen Merkmalsunterraums angibt (k ≤ d)
6. Konstruieren einer Projektionsmatrix *W* aus den *k* in Schritt 5 ausgewählten Eigenvektoren
7. Transformation der *d*-dimensionalen Eingabemenge *X* mit der Projektionsmatrix *W*, um den neuen *k*-dimensionalen Merkmalsunterraum zu erhalten

Für die Implementierung wird auf die optimierte Library von scikit-learn zurückgegriffen:

```
1 pipeline = Pipeline([‘scaling‘, StandardScaler()),
      (‘pca‘, PCA(n_components = 0.95))]
2 X_reduced = pipeline.fit_transform(X)
```

Dabei wird in Zeile 1 eine Pipeline erstellt, in der die Werte zuerst standardisiert werden, bevor der PCA-Algorithmus angewendet wird. Der Parameter *n component* definiert hierbei den prozentualen Anteil an Varianz bei minimal Anzahl Dimensionen, der abgedeckt werden soll. Dieser wird so gesetzt, dass ein ausreichend grosser Anteil der Varianz abgedeckt werden kann. Nach einem Durchlauf der Caltech101-Bilder [16], insgesamt 9144 Bilder in 101 Kategorien, werden 721 Hauptkomponenten erkannt.

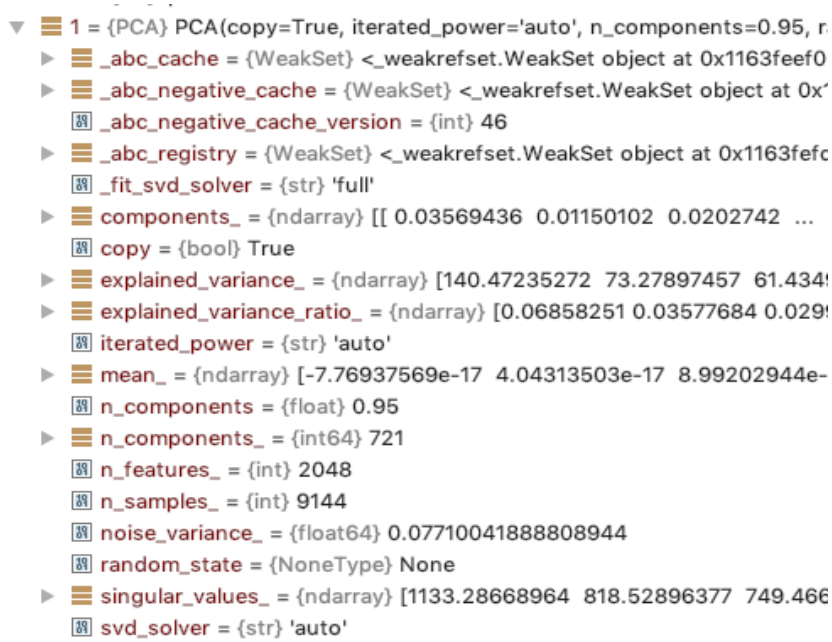

Abbildung 5: PCA-Analyse mit den Caltech101-Bildern

Anmerkung. Eigene Darstellung.

In Abbildung 5 ist beim Parameter *explained variance ratio* exemplarisch zu sehen, dass 0.06858251, also rund 6.9% der Daten, auf der ersten Hauptkomponente liegen, rund 3.6% Prozent auf der zweiten und 2.3% auf der dritten.

3.1.8 Erkenntnisse aus dem Supervised Learning

Obwohl die Kosinus-Ähnlichkeit langsamer rechnet, ist der Zeitverlust gegenüber der Manhattan-Metrik nicht erheblich. Vielmehr kann aufgrund der besseren Klassifizierungsergebnisse an der Kosinus-Ähnlichkeit festgehalten werden. Durch die Reduktion der Dimensionalität könnte die Rechenzeit allenfalls erheblich gesenkt werden. Weitere Tests in diese Richtung wurden nebst der PCA nicht verfolgt.

Das Abschneiden des letzten Layers des Inception-v3 Netzwerkes führte zu besseren Ergebnissen. Daher wurden fortan die Feature-Vektoren verwendet. Mit den Erkenntnissen aus dem Supervised Learning erstellten die Autoren ein Konzept zur Entwicklung eines Prototyps.

3.2 Ein erster grafischer Prototyp

Um möglichst rasch ein Feedback einzuholen und auch eine Bestätigung für den Funktionsumfang der Applikation zu erhalten, wurde mittels Wireframing ein Entwurf erstellt.

3.2.1 Wireframes

Nach Start der Applikation kann entweder mittels eines Suchbegriffes eine Suche gestartet werden oder ein Bild kann als Referenzbild für die Suche importiert werden. Zusätzlich ist die Option “Bild nach dem importieren speichern” anwählbar, womit das importierte Bild auch tatsächlich in der Datenbank persistiert wird. Dies ist in Abbildung 6 ersichtlich.

Abbildung 6: Wireframe – Willkommen

Anmerkung. Eigene Darstellung.

Die gefundenen Suchresultate sollen dann, wie in Abbildung 7 dargestellt, in einer rasterartigen Struktur dargestellt werden. Mittels einer Aktion, wie zum Beispiel einem rechten Mausklick auf ein Bild, erscheint ein Kontextmenu mit den beiden Optionen “Auswählen” und “Bild speichern unter”. Wird die Option “Auswählen” selektiert, so wird das Bild weiter unten in eine Liste angehängt, die als Ausgangslage für eine weitere Suche dient. Alle Bilder, die sich in dieser Liste befinden, werden für eine weitere Suche berücksichtigt.

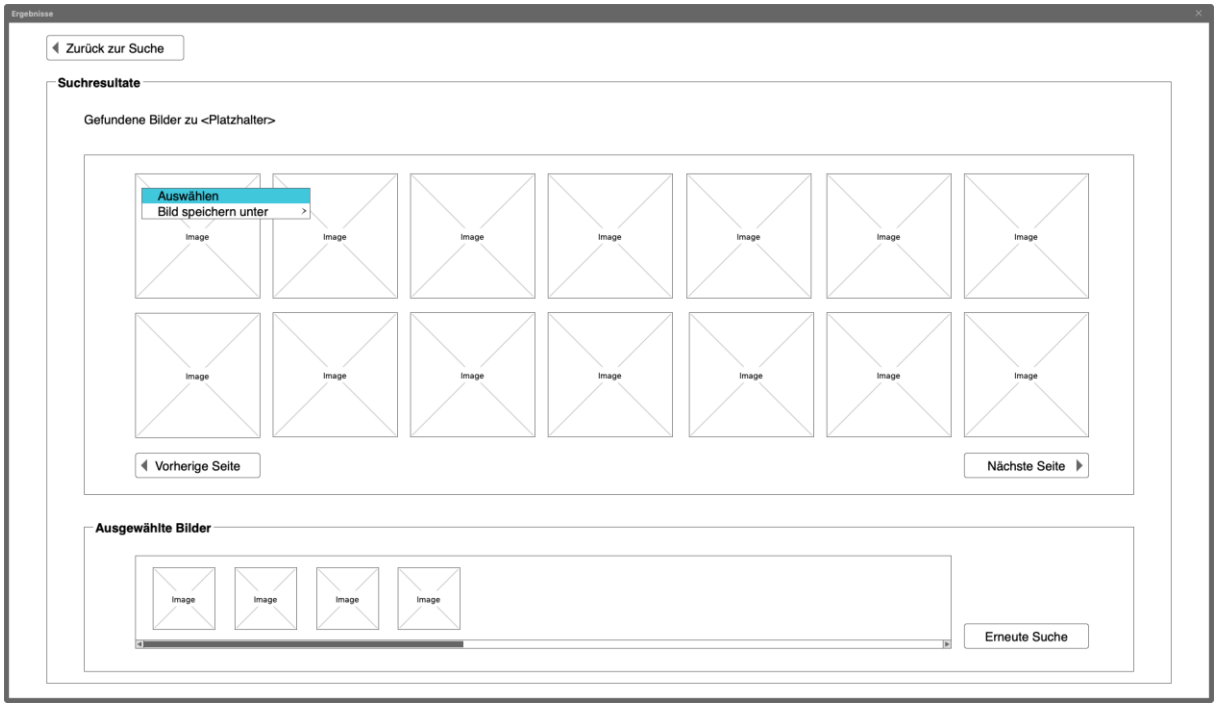

Abbildung 7: Wireframe – Ergebnisse

Anmerkung. Eigene Darstellung.

Mittels einer Aktion, wie beispielsweise einem linken Mausklick auf ein Bild, gelangt man zu einer Detailansicht. Das in Abbildung 8 ersichtliche Wireframe illustriert eine mögliche Detailansicht.

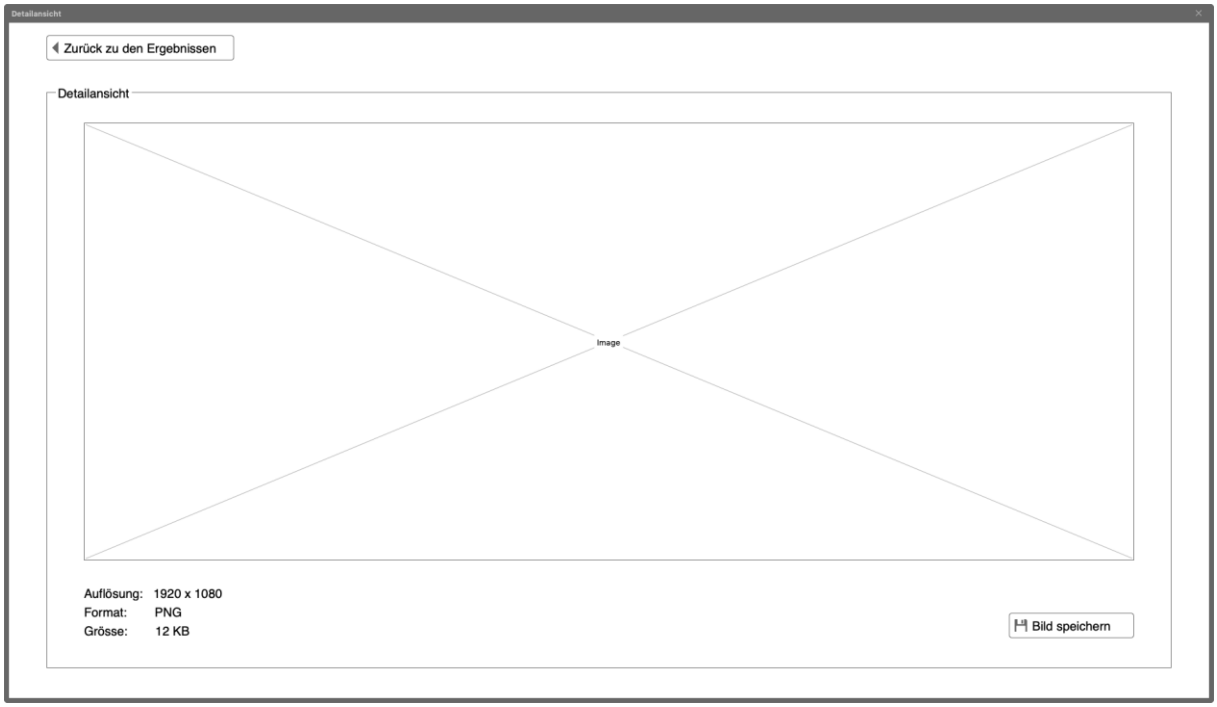

Abbildung 8: Wireframe – Detailansicht

Anmerkung. Eigene Darstellung.

In dieser Ansicht lässt sich ein Bild vergrössert betrachten und es werden Metainformationen wie Auflösung, Format und Grösse, angezeigt.

3.2.2 Auswertung der Wireframes

Das Feedback wurde von der Begleitperson eingeholt. Dabei hat sich herausgestellt, dass eine Benutzeroberfläche vergleichbar mit der Benutzeroberfläche der Google-Suche wie sie in Abbildung Abbildung 9 zu sehen ist, bevorzugt wird.

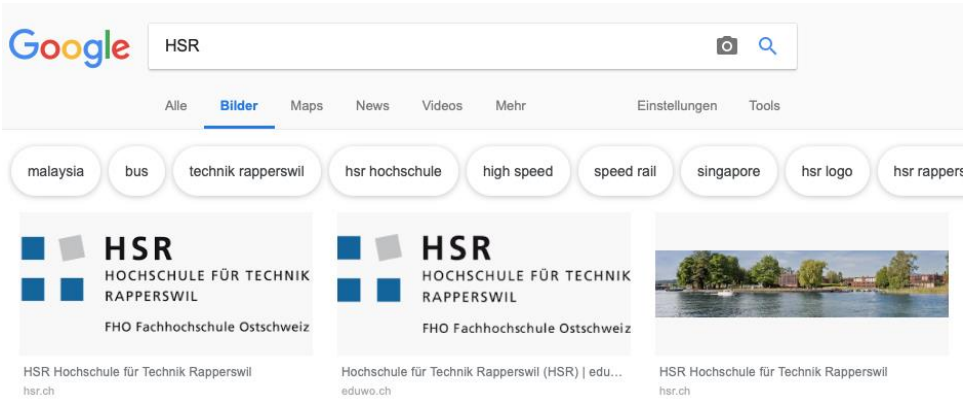

Abbildung 9: Print-Screen der Google-Bildersuche

Anmerkung. Eigene Darstellung.

Somit können die in Abbildung 6 und Abbildung 7 ersichtlichen Funktionalitäten, in einem Fenster zusammengefasst dargestellt werden. Ebenfalls entfallen die in Abbildung 8 enthaltenen Buttons “Vorherige Seite” und “Nächste Seite” die für die Anzeige der nächsten Resultate zuständig sind, da die Oberfläche durch eine Endlosliste mit Scroll-Funktion ersetzt wird. Um die Benutzerfreundlichkeit zu optimieren, wurde statt der in Abbildung 7 angezeigten Option für die Auswahl eines Bildes, eine Drag-and-Drop Funktion vorgeschlagen. Damit kann ein Bild direkt in die Liste der ausgewählten Bilder verschoben werden. Nachfolgend werden die wichtigsten Kriterien für die Evaluation eines geeigneten Frameworks erläutert.

Drag-and-Drop: Es hat sich herausgestellt, dass dieses Feature einen grossen Einfluss auf die Benutzerfreundlichkeit hat. Deshalb wird es in die Bewertung miteinbezogen.

Dokumentation: Damit bei der Entwicklung mehr als nur von Beispielen kopierte Applikationen entstehen können, ist eine gute Dokumentation unabdingbar.

Erprobtes Framework: Bei diesem Kriterium spielt es vor allem eine Rolle, wie lange das Framework bereits auf dem Markt existiert. Eine Version vor 1.0 kann unter Umständen fehlerhaft sein. Auch sind Alphaversionen in der Regel nicht sehr funktionsreich.

Einfachheit: Bei der Einfachheit spielt es vor allem eine Rolle, wie schnell man zu einer lauffähigen Applikation kommt und wie umständlich es ist, Widgets und Events zu implementieren.

3.2.3 Bewertung möglicher Frameworks

In Tabelle 8 wurden mögliche Frameworks anhand der definierten Kriterien in einem Bewertungsraster auf einer Skala von 1 bis 6 durch die Autoren bewertet.

Tabelle 8: Bewertung ausgewählter Frameworks

Name	Drag & Drop	Dokumentation	Erprobtes Framework	Einfachheit	Total	Bemerkungen
appJar [23]	4	5.5	4	5.5	19	Baut auf Tkinter auf
PyQt [24]	5.5	5	6	4.5	21	Python Variante des aus C++ bekannten QT Frameworks
PySimpleGUI [25]	1	5	4.5	6	16.5	Baut auf Tkinter auf
Tkinter [26]	4	4.5	6	4	18.5	Gehört zum Lieferumfang von Python
Durchschnitt	3.6	5	5.1	5	18.8	

Anmerkung. Eigene Darstellung.

3.2.4 Auswertung möglicher Frameworks

Die Anforderungen des Anwendungsbereichs der Evangelisch-reformierten Kirche Horgen werden von PyQt5 am besten erfüllt. Es bietet als einziges Framework die Drag-and-Drop-Funktionalität an, verfügt über eine gute Dokumentation und ist schon seit 1995 auf dem Markt [27]. Obwohl Tkinter zum Lieferumfang von Python gehört, erscheint die Dokumentation im Vergleich minimalistisch. Tkinter gehört zum Lieferumfang von Python und bietet daher den Vorteil, dass es aufgrund der grossen Verbreitung industrieerprobt ist. Dies zeigt sich auch darin, dass nun einige Frameworks wie PySimpleGUI und appJar auf Tkinter aufbauen. Diese Frameworks versuchen die Schwächen in der Dokumentation und der Einfachheit von Tkinter zu minimieren. Grundsätzlich wäre auch der Einsatz von appJar denkbar gewesen. Da jedoch erst die Version 0.93 veröffentlicht wurde und sich die Drag-and-Drop-Funktionalität noch in einer Betaversion befindet, scheint es den Autoren noch zu früh, mit diesem Framework zu arbeiten. Daher entschieden sich die Autoren PyQt5 zu verwenden.

3.2.5 Architektur

Die Applikation ist wie in Abbildung 10 dargestellt, in neun Module gegliedert.

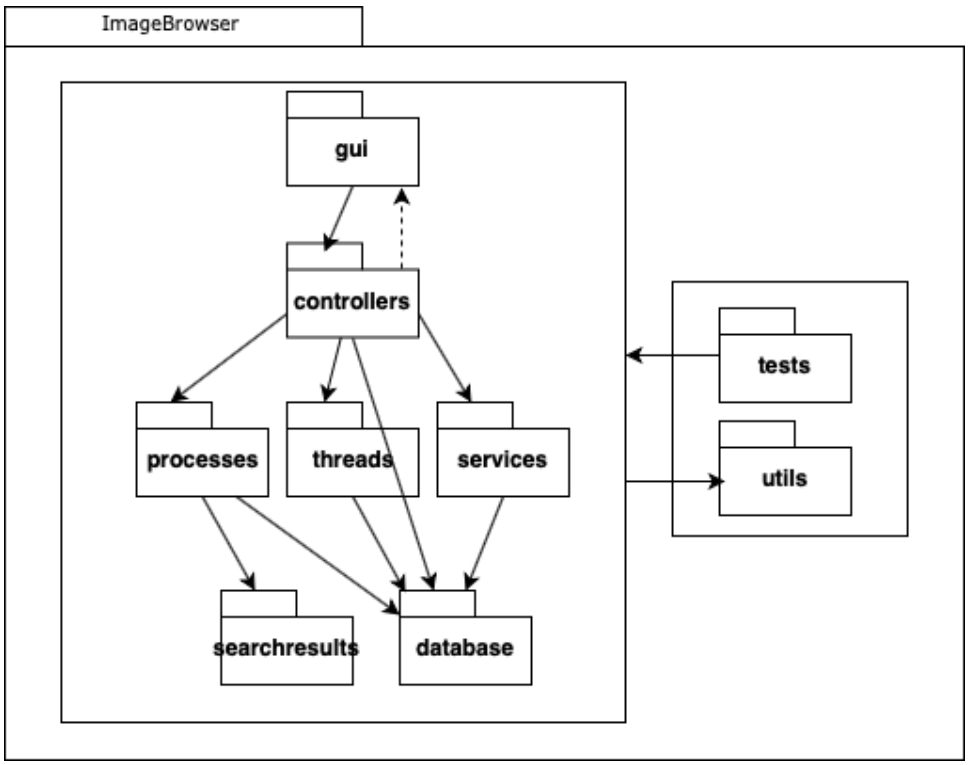

Abbildung 10: Package Diagramm

Anmerkung. Eigene Darstellung.

Dabei greifen weiter oben angesetzte Module jeweils auf weiter unten angesiedelte Module zu. Vom Graphical User Interface (GUI) werden jeweils Events ausgelöst, die den Controllern weitergegeben werden. Die Controller selbst können nur zur Laufzeit das GUI benachrichtigen. Dies ist in Abbildung 10 mit einem gestrichelten Pfeil symbolisiert. Die Controller können jeweils eine Stufe überspringen, wenn sie nur lesend zugreifen, was zum Beispiel beim Start der Applikation der Fall ist. Das Modul “utils” enthält einige Hilfsfunktionen und wird von vielen Modulen genutzt. Processes, Threads und Services werden jeweils verwendet, um gewisse Abläufe, wie das Kopieren von Files, asynchron abzuarbeiten.

3.3 Referenzbilder für die lokale Suche

Um an ein Referenzbild für die lokale Suche zu gelangen, bietet sich eine Abfrage bei einer Suchmaschine an. Nachfolgend werden in Tabelle 9 drei Varianten auf ihre Eignung geprüft. Dabei wird die Qualität der erhaltenen Abfragen nicht berücksichtigt.

Tabelle 9: Vergleich von Möglichkeiten zur Abfrage von Bildern

	Google Custom Search [28]	**Microsoft Bing Image Search API [29]**	**google-images-download [30]**
Kosten	100 Abfragen / Tag kostenlos, danach $5 pro 1'000 Abfragen	3'000 Abfragen / Monat kostenlos, danach $7 pro 27'000 Abfragen	kostenlos
REST-API	Ja	ja	nein
Python Client-Library	Ja	ja	ja
Dokumentation	Befriedigend	sehr gut	sehr gut
Einfachheit	Umständlich	sehr einfach	sehr einfach
Bemerkungen	Google bietet zwar eine Library an, jedoch ist der Prozess der Authentifizierung eher schwerfällig ausgefallen.	Die Abfrage enthält an erster Stelle die URL's der Treffer; das heisst, dass zu jedem Treffer jeweils noch eine weitere Abfrage benötigt wird.	Open Source Projekt mit über 2'500 Likes auf GitHub; Einschränkung: 100 Bilder pro Abfrage.

Anmerkung. Eigene Darstellung.

Sowohl Google als auch Microsoft bieten mittlerweile gute Schnittstellen für das Abfragen von Bildern an. Es zeigt sich, dass die Nutzung von Google für Entwickler umständlicher ist im Vergleich zu anderen Anbietern. So sind zusätzliche Abläufe zur Authentifizierung notwendig und die Dokumentation für die Python Client-Library fällt mit einem kurzen Beispiel sehr schlank aus [31]. Die Autoren gehen davon aus, dass der Zugriff auf gewisse Google-Dienste einheitlich gestaltet werden und es deshalb mit Mehraufwand verbunden ist.

Microsoft sticht mit einer sehr einfachen und soliden Dokumentation hervor. Zu einer Abfrage werden lediglich URL's und weitere Metadaten der Treffer zurückgegeben, was bedeutet, dass für jeden Treffer jeweils noch ein separater Aufruf notwendig ist, damit man letztendlich das Bild erhält.

Auf GitHub hat sich für das Herunterladen von Bildern bei Google eine Python-Library durchgesetzt, die einen interessanten Ansatz verfolgt.

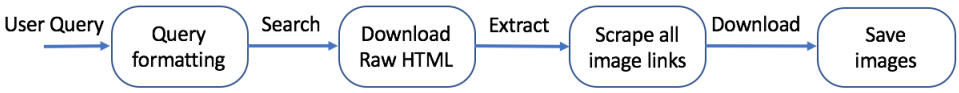

Abbildung 11: Algorithmus zum Download der Google-Bilder

Anmerkung. Die Abbildung bezieht sich auf die Quelle [31]

Wie in Abbildung 11 dargestellt, wird eine Abfrage zuerst von der Library formatiert, bevor diese an Google gesendet wird. Als Resultat der Anfrage wird ein HTML-File von Google heruntergeladen und vorprozessiert. Von diesem File werden anschliessend alle URL extrahiert und in einem weiteren Schritt die effektiven Bilder heruntergeladen. So wird ein allfälliger Authentifizierungsprozess, wie bei Verwendung der Google Custom Search, umgangen und die Anzahl Abfragen sind so nicht limitiert. Dieses Verfahren bringt aber den Nachteil mit sich, dass ein gewisser Overhead entsteht und so mehr Daten konsumiert werden.

Fazit
Die Autoren haben sich für die Verwendung der google-images-download-Bibliothek entschieden, da sie keine Restriktionen in den Anzahl Abfragen vorweist und durch die simple Anwendung hervorsticht. Auch die Implementation der Konkurrenten von Microsoft und Google wäre möglich gewesen. Der Nachteil des Overheads kann jedoch in der heutigen Zeit ignoriert werden. Es haben sich bei Tests keine negativen Auswirkungen auf die Performanz gezeigt.

4 Umsetzung

Dieses Kapitel beschreibt den Funktionsumfang der fertigen Applikation.

4.1 Funktionalitäten des Prototyps

Die folgenden Beschreibungen beziehen sich auf den Screenshot der Abbildung 12. Beschrieben wird der Ablauf, wie Bilder in die Applikation importiert werden und anschliessend eine Suche nach ähnlichen Bildern durchgeführt werden kann.

- Als erstes muss die gewünschte Bilddatenbank über den Button „3 Bildimport“ importiert werden. Ist dieser Prozess abgeschlossen, kann die Suche über die Bilddatenbank starten.

- Die Applikation wurde mit einer Google-Bildersuche ausgestattet, welche es erlaubt, Bilder von Google herunter zu laden und aus der lokalen Datenbank ähnliche Bilder wie das Google-Bild zu suchen. Dazu wird ein Suchbegriff im Feld „1 Bildsuche“ eingegeben. Anschliessend wird die Suche mit dem Button "Suche Google Bild" oder durch Betätigen der Taste „Enter“ gestartet. Die bei Google gefundenen Bilder erscheinen anschliessend in Feld „4 Suchresultate“. Alternativ kann ein Bild aus dem lokalen Filesystem per Drag & Drop in das Eingabefeld „1 Bildsuche“ gezogen werden. Durch Drücken der Enter-Taste wird das Bild ebenfalls in die Liste „4 Suchresultate“ aufgenommen.

- Über den Button „6 Bilder entfernen“ können Bilder aus der Liste „4 Suchresultate“ wieder entfernt werden.

- Startet man einen Suchvorgang nach ähnlichen Bildern in der lokalen Bilddatenbank, erscheinen anschliessend die Treffer im Feld „2 Ergebnisliste“. Wenn sich mehrere Bilder in der Liste "4 Suchresultate” befinden, wird der arithmetische Mittelwert der Bilder berechnet und für die Suche nach ähnlichen Bildern verwendet. Diesbezüglich wird auf den Abschnitt 5.1.3 verwiesen.

- Sobald der Suchvorgang nach ähnlichen Bildern beendet ist, erscheinen in der "2 Ergebnisliste" die ähnlichsten Bilder aus der Datenbank. Die Treffer werden automatisch absteigend nach Ähnlichkeit sortiert.

- Die in der „2 Ergebnisliste“ angezeigten Bilder können per Drag-and-Drop in die „4 Suchresultate“ Spalte gezogen werden. Dies ermöglicht eine Suche mit Bildern der eigenen Datenbank.

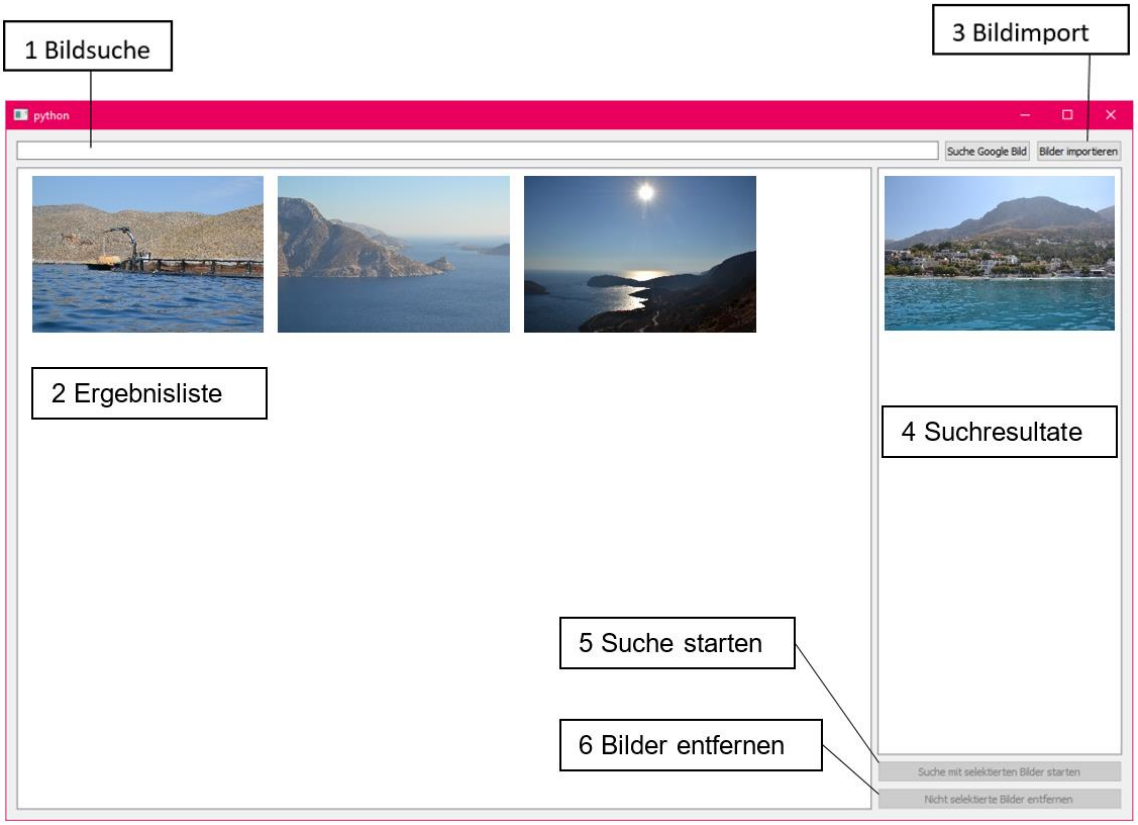

Abbildung 12: Screenshot des Prototyps

Anmerkung. Eigene Darstellung.

4.2 Clustering

Ziel des Clusterings war es, einen Überblick über die komplette Bildersammlung zu erhalten. Nach Start der Applikation sollten möglichst unterschiedliche Bilder der Bildersammlung in der Ergebnisliste angezeigt werden.
Die Wahl des Clustering Verfahrens wurde aufgrund folgender Anforderungen ausgewählt:

- Anzahl der Cluster nicht bekannt
- Erste Cluster sollen möglichst schnell berechnet werden können
- Nachladen weiterer Cluster möglich

Aufgrund der Anforderungen wurde ein an das Konzept des hierarchischen Clusterings mit Top-Down-Verfahren [32] gewählt. Nachfolgend wird die Umsetzung genauer erläutert.

4.2.1 Gleichheitsmatrix

In einer ersten Phase wurde mit der sklearn.metrics.pairwise.cosine_similarity() Methode von scikit [33] eine Gleichheitsmatrix erstellt. Die Methode berechnet für sämtliche mögliche Bilderpaare die Distanz und speichert diese in einer Matrix. Eine exemplarische Darstellung des Resultats ist in Tabelle 10 ersichtlich.

Tabelle 10: Exemplarische Darstellung der Gleichheitsmatrix

	Bild 1	**Bild 2**	**Bild 3**	**Bild 4**
Bild 1	1	0.2	0.5	0.8
Bild 2	0.2	1	0.3	0.7
Bild 3	0.5	0.3	1	0.9
Bild 4	0.8	0.7	0.9	1

Anmerkung. Eigene Darstellung.

Aus der generierten Matrix war es nun möglich den „min“ Value, also das Bildpaar, welches die geringste Ähnlichkeit zueinander aufweist, herauszulesen. Numpy bietet dazu eine entsprechende Funktion, numpy.matrix.min() [34] , an. Als Beispiel wäre dies in Tabelle 10 der Wert 0.2. Dies bedeutet, dass Bild 1 zu Bild 2 die geringste Ähnlichkeit aufweist. Alle anderen Bilder weisen einen höheren Wert auf und sind daher ähnlicher zueinander.

4.2.2 Auswahl der Bilder für den Cluster

Die vorgängig eruierten Bilder 1 und 2 werden als Bilder für den Cluster vorgemerkt und in einer Liste gespeichert. Die aktuelle Liste der Bilder für den Cluster ist in Tabelle 11 ersichtlich.

Tabelle 11: Exemplarische Darstellung der Liste mit Clusterbilder

Bild 1	Bild 2		

Anmerkung. Eigene Darstellung.

Der gefundene Minimalwert 0.2 wird nun in der Matrix auf 1 gesetzt, damit die numpy.matrix.min() Funktion diesen nicht mehr findet. Das Ergebnis ist in Tabelle 12 dargestellt.

Tabelle 12: Exemplarische Darstellung der Gleichheitsmatrix nach 1. Iteration

	Bild 1	**Bild 2**	**Bild 3**	**Bild 4**
Bild 1	1	1	0.5	0.8
Bild 2	1	1	0.3	0.7
Bild 3	0.5	0.3	1	0.9
Bild 4	0.8	0.7	0.9	1

Anmerkung. Eigene Darstellung.

Nun kann erneut der kleinste Wert aus der Matrix herausgelesen werden. Es ist allerdings nicht ausreichend, lediglich die kleinsten Werte aus der Matrix auszulesen und die zugehörigen Bilder dem Cluster hinzuzufügen. Die damit generierte Liste mit Bildern garantiert nicht, dass die Bilder untereinander ebenfalls genügend ungleich sind. Nach dem Hinzufügen der ersten beiden Bilder muss zusätzlich geprüft werden, ob die weiteren Bilder bereits in der Liste der Clusterbilder enthalten sind und das Bild zu sämtlichen Bildern in der Liste der Clusterbilder eine genug hohe Distanz aufweist. In einem nächsten Schritt wird, wie in Tabelle 12 ersichtlich, als nächst kleinerer Wert 0.3 genommen. Für die dazugehörigen Bilder, Bild 2 und Bild 3, wird nun geprüft, ob diese bereits in der Liste der Clusterbilder enthalten sind und ob die Bilder zu sämtlichen Bildern in der Liste der Clusterbilder eine genug hohe Distanz aufweisen. Bild 2 ist bereits in der Liste und wird folglich ignoriert. Bild 3 bleibt als möglicher Kandidat bestehen und muss mit Bild 1, welches sich in der Liste der Clusterbilder befindet, auf Gleichheit überprüft werden. Darauf wird im nachfolgenden Abschnitt eigegangen. Ein Ausschnitt des Programmcodes für die Auswahl der Clusterbilder ist nachfolgend ersichtlich.

```
while len(hits) < configuration_constants.number_of_clusters_to_show_in_ui(
    ) and min_value_in_matrix < configuration_constants.threshold():
        min_value_in_matrix = similarity_matrix.min()
        position_y_x = np.where(similarity_matrix == min_value_in_matrix)
        y = position_y_x[0][0]
        x = position_y_x[1][0]
        y_similar = False
        x_similar = False
        for array in hits:
            if array[y] > configuration_constants.threshold():
                y_similar = True
            if array[x] > configuration_constants.threshold():
                x_similar = True
        if not y_similar:
            hits.append(similarity_matrix[y])
            clustering.append(keys[y])
        if not x_similar:
            hits.append(similarity_matrix[x])
            clustering.append(keys[x])
```

4.2.3 Threshold aus der ROC-Kurve als Distanzschwelle

Für die Entscheidung, ob die Distanz zwischen Bild 3 und Bild 1 ausreichend gross ist, wurde ein Threshold Wert aus der in Abbildung 13 zu sehenden, grünen ROC-Kurve als Entscheidungsschwelle definiert. Um den Threshold Wert zu berechnen, wurde derjenige Punkt auf der grünen Linie fixiert, welcher zum Punkt (0, 1) die kleinste Distanz aufweist. Dies garantiert, dass bei einer Vergleichsweise niedrigen False Positive Rate die True Positive Rate hoch ist. Die Distanz ist in Abbildung 13 mit einem blauen Pfeil gekennzeichnet.

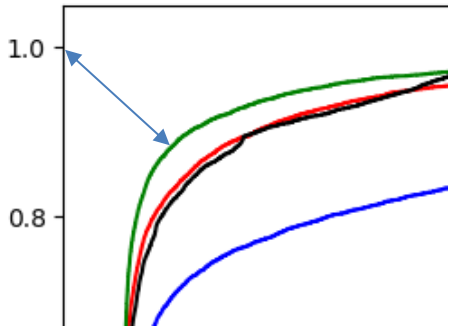

Abbildung 13: Kleinste Distanz zum Punkt (0,1)

Anmerkung. Eigene Darstellung.

Die minimale Distanz zum Punk (0,1) wurde mit der Pythagoras Formel ermittelt. Folglich konnte ein Punkt auf der Kurve fixiert werden. Aufgrund der durch den Punkt erhaltenen True und der False Positive Rate konnte mit der „roc_curve“ Methode on sklearn.metrics der zugehörige Schwellwert 0.569 eruiert werden.

Wie in Tabelle 12 dargestellt ist, weist die Gleichheit zwischen Bild 3 und Bild 1 einen Wert von 0.5 auf. Da 0.5 unter der Entscheidungsschwelle von 0.569 liegt, kann somit auch Bild 3 zur Liste der Clusterbilder hinzugefügt werden, was in Tabelle 13 ersichtlich ist.

Tabelle 13: Vollständige Liste mit den Clusterbildern

Bild 1	Bild 2	Bild 3	

Anmerkung. Eigene Darstellung.

Dieses Vorgehen kann so lange wiederholt werden, bis die gewünschte Anzahl der Cluster erreicht oder keine geeigneten Bilder mehr gefunden werden. Die in der Liste gespeicherten Bilder werden anschliessend in der Applikation angezeigt, was in Abbildung 12, „2 Ergebnisliste“, ersichtlich ist.

5 Test

In diesem Kapitel wird auf die durchgeführten Tests der entwickelten Architektur eingegangen. Weiter wird erläutert, ob die definierten funktionalen sowie nicht funktionalen Anforderungen umgesetzt werden konnten.

5.1 Unsupervised Learning Tests

Da feststand, dass die Feature-Vektoren in Kombination mit der Kosinus-Ähnlichkeit verwendet werden sollten, wurden schliesslich weitere Tests mit den Bildern der Evangelisch-reformierten Kirche Horgen durchgeführt. Dazu wurden sämtliche aus den 15'295 Bilder berechneten Feature-Vektoren auf das Filesystem gespeichert. Die Bilder konnten nun mit dem siamesischen Netzwerk verglichen werden. Da die Bilder der Evangelisch-reformierten Kirche Horgen nicht klassifiziert sind, mussten die Ergebnisse visuell geprüft werden.

5.1.1 Datenschutz

Es können in dieser Dokumentation aus Datenschutzgründen keine Bilder der Evangelisch-reformierten Kirche Horgen gezeigt werden, welche Personen enthalten. Da dies auf einem Grossteil der Bildaufnahmen zutrifft, werden die Ergebnisse der Tests mit Hilfe von Balkendiagrammen festgehalten. In Abbildung 23 (Anhang F) befinden sich zur Illustration drei Bildersuchresultate ohne Personen, die zur Veröffentlichung freigegeben wurden. Für das Erstellen der Diagramme wurden die 15'295 Bilder grob durchgesehen und inhaltlich einige zufällige Bildmerkmale ausgewählt. Zu diesen Bildmerkmalen wurden anschliessend Suchläufe mit einem passenden Referenzbild von Google gestartet.

5.1.2 Suche nach Merkmalen in den Bildern

Während den Testläufen wurde versucht, die Grenzen des Netzwerkes zu eruieren. Dazu wurden die Anforderungen an ein Suchresultat, beziehungsweise die vorgegebenen Merkmale eines Bildes, stetig erhöht.

Suche nach einem Merkmal

Wie in Abbildung 14 zu entnehmen ist, wurden zuerst Tests mit einem Bildmerkmal durchgeführt. Das heisst, dass ein passendes Bild von Google gesucht wurde, welches zum Beispiel eine Blume darstellt. Weitere Merkmale zur Blume wurden vorerst nicht berücksichtigt. Anschliessend wurde die Suche gestartet und überprüft, wie viele der zehn von der Applikation erhaltenen Ergebnisbilder, das Merkmal Blume enthielten. In insgesamt 20 Testdurchläufen wurden im Durchschnitt 8.45 von zehn korrekten Bildern angezeigt.

Um ein möglichst aussagekräftiges Ergebnis zu erhalten, wurde in der Bildersammlung sichergestellt, dass mindestens zehn passende Bilder vorhanden sind. Dabei ist zu erwähnen, dass die Anzahl passender Bilder in der Sammlung einen Einfluss auf das Ergebnis hat. Befinden sich beispielsweise 100 statt 10 passende Bilder für das Merkmal Blume in der Sammlung, ist die Chance grösser, passende Bilder zu finden. Diesen Umstand zu berücksichtigen und somit die Bilder einzeln zu klassifizieren, hätte den Rahmen dieser Studienarbeit gesprengt und wurde somit vernachlässigt.

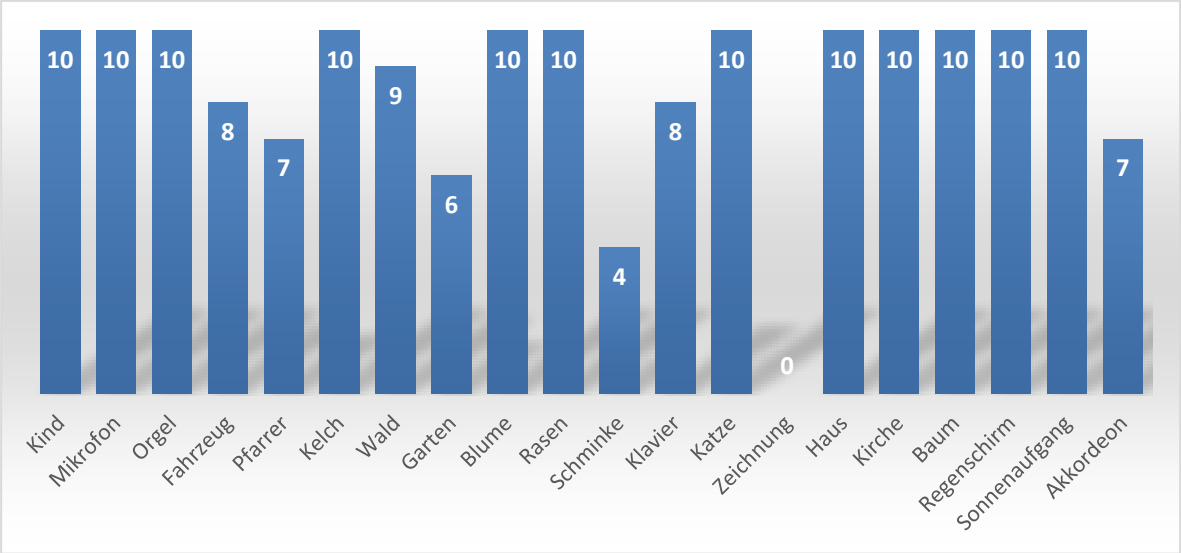

Abbildung 14: Treffer anhand Google Suchbilder

Anmerkung. Eigene Darstellung.

Die in Abbildung 14 ersichtlichen Ergebnisse zeigen, dass Bilder mit allgemeinen oder abstrakten Begriffen, wie beispielsweise Schminke oder Zeichnung, ungenügende Ergebnisse liefern. Weiter unterscheidet sich ein Pfarrer von einer normalen Person lediglich durch das Kollar. Dies war für das Netzwerk schwer zu unterscheiden. Das bei der Suche nach einem Akkordeon lediglich sieben Treffer gefunden wurden, könnte darauf zurückgeführt werden, dass nicht wesentlich mehr als 10 Bilder mit Akkordeons in der Bildersammlung vorhanden sind.

Suche nach mehreren Merkmalen

Anschliessend wurden in der Datenbank Bilder mit mehreren Merkmalen zufällig ausgewählt. Beispielsweise wurde nicht nur darauf geachtet, dass eine Blume im Bild enthalten ist, sondern auch ob diese Blume in einer Vase steht oder nicht. Da mit zunehmenden Merkmalen die Anzahl passender Bilder in der Datenbank abnimmt, wurde die Suche bereits ab einem Treffer als erfolgreich gewertet. Die definierten Merkmale der einzelnen Bilder sind in Abbildung 15 ersichtlich.

Nun wurde bei Google wiederum ein zu den Merkmalen passendes Bild gesucht. Die Bilder von Google wurden als Referenzbilder definiert und eine Suche gestartet. Wurde ein Bild mit den Merkmalen bei den Ergebnissen angezeigt, galt die Suche als erfolgreich. Die Ergebnisse der Vergleiche können aus Abbildung 15 entnommen werden. Zeigt die Achse eine 1 an, wurde ein Bild mit den speziellen Merkmalen gefunden. Zeigt die Achse jedoch eine 0 an, wurde kein Bild mit den Merkmalen gefunden.

Der Versuch zeigte, dass insgesamt 14 von 20 Suchvorgängen erfolgreich waren. Bei diesem Versuch spielt die Anzahl Bilder in der Bildersammlung, welche nur eines der Merkmale enthalten, eine entscheidende Rolle. Ein Beispiel dafür ist die Suche nach einem Kind mit herausgestreckter Zunge. Die Bildersammlung enthält sehr viele Kinder. Die Suche zeigte zehn Bilder mit Kindern an, jedoch keines mit herausgestreckter Zunge. Trotzdem kann der Test mit 14 von 20 richtigen Ergebnissen als positiv gewertet werden.

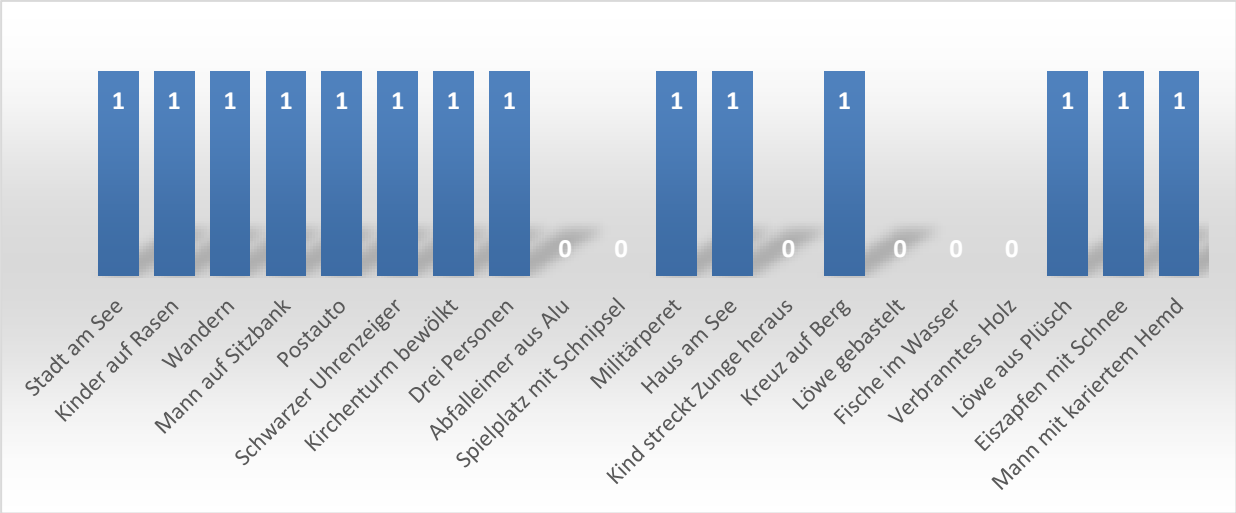

Abbildung 15: Treffer nach Google Suchwort spezifisch

Anmerkung. Eigene Darstellung.

Verfeinerte Suche nach mindestens drei Merkmalen

Anschliessend wurden die Suchbegriffe weiter verfeinert und wiederum geprüft, ob ein entsprechendes Bild gefunden wurde. Die Resultate dieser Suchvorgänge sind in Abbildung 16 ersichtlich. Das Ergebnis zeigte, dass vor allem die Suche nach einer Person mit bestimmten Merkmalen ungenügende Ergebnisse lieferte. Objekte wurden grundsätzlich besser erkannt.

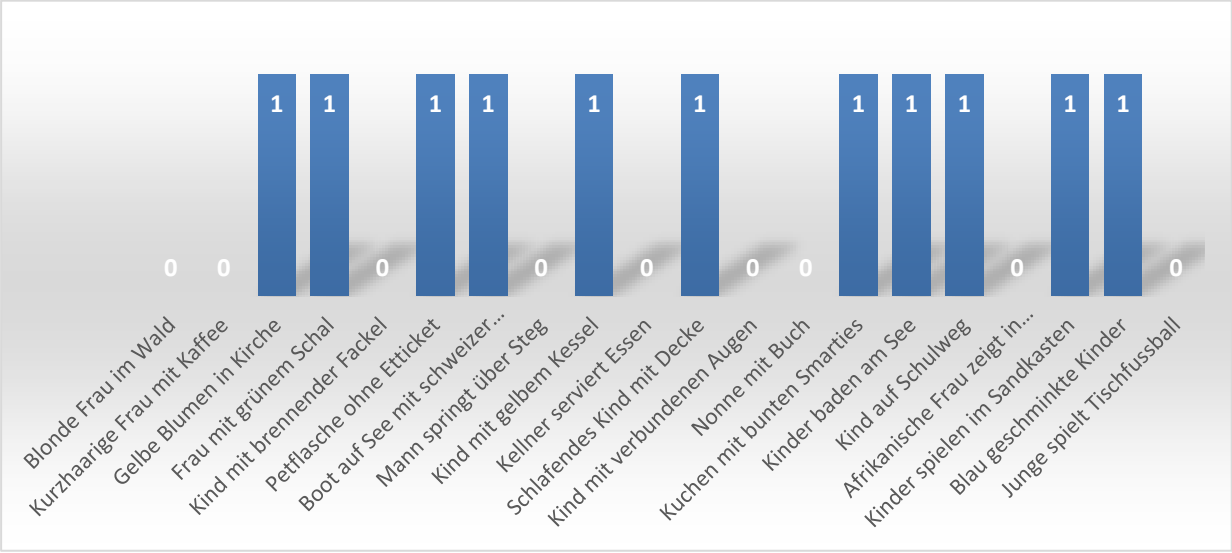

Abbildung 16: Treffer nach Google Suchwort sehr spezifisch

Anmerkung. Eigene Darstellung.

Erkenntnisse aus der Suche mit Merkmalen

Bildvergleiche mit den Feature-Vektoren und der Kosinus-Ähnlichkeit lieferten für die Studienarbeit ausreichend gute Ergebnisse. Werden die Merkmale innerhalb der Bilder spezifischer,

vor allem beim Hinzufügen von Merkmalen oder Eigenschaften zu Personen, zeigten die Resultate abnehmende Qualität an. Unterschiede zwischen einer erwachsenen Person und einem Kind oder einem Jungen oder einem Mädchen, werden oft erkannt. Eine bestimmte Person zu finden war jedoch nur zufällig möglich.
Es zeigte sich auch, dass das Netzwerk Eigenschaften wie Farben oder Helligkeit nicht berücksichtigt. Daher führten Suchvorgänge nach Bilder, welche Stimmungen oder Gefühle ausdrücken, zu keinem positiven Ergebnis.

5.1.3 Suche mit mehreren Referenzbildern

Die entwickelte Applikation unterstützt die Suche über mehrere Referenzbilder. Enthält ein Referenzbild beispielsweise einen Hund und das andere eine Katze, sollen Bilder, welche gleichzeitig einen Hund und eine Katze enthalten, angezeigt werden. Um diese Funktionalität umzusetzen wurden diverse Versuche durchgeführt.

Referenzbilder iterativ abarbeiten

Als erster Entwurf wurde jedes Referenzbild mit allen Bildern aus der Bildersammlung der Evangelisch-reformierten Kirche Horgen verglichen. Dabei wurden die erhaltenen Ähnlichkeiten addiert und gespeichert. Das Vorgehen ist in Tabelle 14 visualisiert.

Tabelle 14: Iterative Ähnlichkeitsbeurteilung

	Bild 1	**Bild 2**	**Bild 3**	**Bild 4**
Referenzbild 1	0.5	0.4	0.8	0.7
Referenzbild 2	0.2	0.9	0.4	0.3
Summe der Ähnlichkeiten	0.7	**1.3**	1.2	1.0
Bester Treffer		x		

Anmerkung. Eigene Darstellung.

Dieses Vorgehen evaluiert die relevanten Features beider Referenzbilder. Der Nachteil dieser Vorgehensweise ist, dass sich mit jedem zusätzlichen Referenzbild die Anzahl Distanzberechnungen verdoppelt. Wie aus Abbildung 17 entnommen werden kann, beträgt die reine Berechnungszeit bei zehn Referenzbildern 7.02 Sekunden. Diese Berechnungszeit erscheint als zu lange, um mit der Applikation arbeiten zu können.

Arithmetisches Mittel aus Referenzbildern

Ein anderer Ansatz war aus allen Vektoren der Referenzbilder den arithmetischen Mittelwert zu berechnen und diesen einmal mit den Bildern aus der Bildersammlung zu vergleichen. Dadurch muss nur die Zeit zur Berechnung des arithmetischen Mittelwertes zur Gesamtzeit dazugezählt werden. SciPy bietet mit numpy.mean() [35] eine Methode an, mit welcher die Autoren Performanz- und Qualitätstests durchgeführt haben. Da die Berechnung des Mittelwertes für 10 Bilder bei 0.0001 Sekunden lag wurde die Berechnungszeit vernachlässigt. Das Ergebnis ist in Abbildung 17 ersichtlich. Die X-Achse zeigt dabei die Anzahl Referenzbilder an und die Y-Achse die Benötigte Zeit in Sekunden.

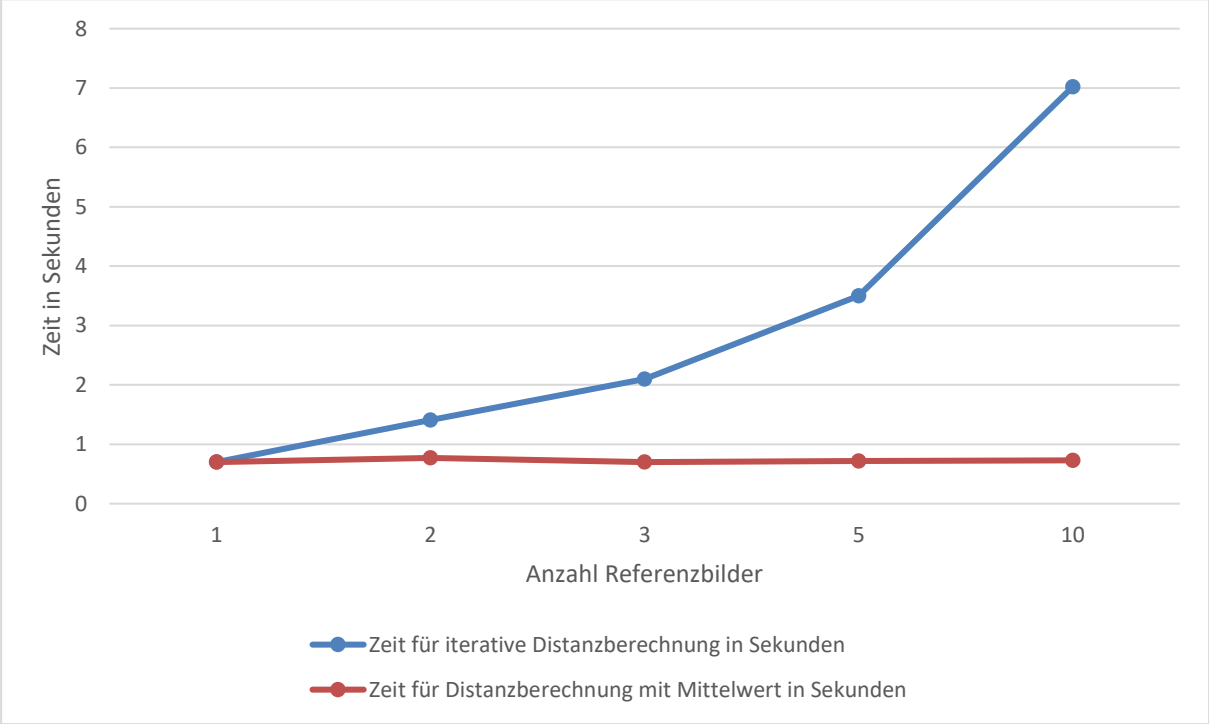

Abbildung 17: Unterschied iterativ und arithmetischer Mittelwert

Anmerkung. Eigene Darstellung.

Aus Abbildung 17 ist zu entnehmen, dass die Berechnungszeit mit dem arithmetischen Mittelwert im Gegensatz zur iterativen Berechnung linear verläuft. Mit dem iterativen Verfahren werden für die Berechnung mit 10 Referenzbildern bereits 7 Sekunden benötigt. Daher wurde eine Umsetzung mit dem arithmetischen Mittelwert angestrebt.

Im Anschluss wurden mit der Bildersammlung der Evangelisch-reformierten Kirche Horgen Tests mit zwei Referenzbildern durchgeführt. Das Ergebnis kann aus Abbildung 18 entnommen werden.

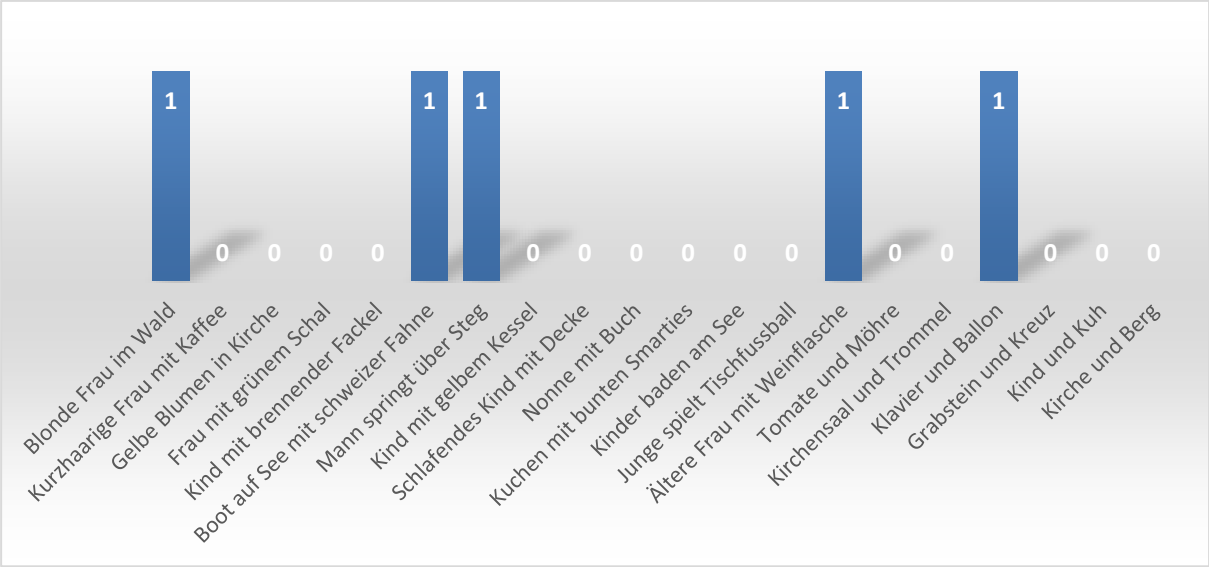

Abbildung 18: Ergebnis Suche mit zwei Referenzbildern

Anmerkung. Eigene Darstellung.

Von den durchgeführten Tests fielen 15 von 19 negativ aus. Zum Teil wurde das Ergebnis sogar verschlechtert. Bei der Suche nach „Trommel“ und „Kirchensaal“ fand sich kein passender Treffer in der Auswahl. Wird jedoch nur nach „Trommel“ gesucht, zeigt sich ein gültiger Treffer. Vereinzelt wurden positive Ergebnisse verzeichnet. Beispielsweise ergab die Suche nach „Klavier“ und „Ballon“ mehrere gültige Treffer, welche mit nur einem Referenzbild (Ballon oder Klavier), nicht angezeigt wurden.

Zu den vier erfolgreichen Tests ist zu erwähnen, dass bei der Suche mit nur einem der beiden Referenzbilder kein gültiges Bild gefunden wurde. So konnte der Einfluss auf das Ergebnis durch den arithmetischen Mittelwert nachvollzogen werden.

Schlussfolgerung zur Suche mit mehreren Referenzbildern

Obwohl viele Tests mit mehreren Referenzbildern fehlschlugen, kann diese Art der Suche in einigen Fällen zielführend sein. Eine Optimierung sollte jedoch in Betracht gezogen werden. Ansätze dazu werden im Kapitel 8 „Zusammenfassung und Ausblick“ erläutert.

5.2 Blackbox-Test der funktionalen Anforderungen

In Tabelle 3 wurden die funktionalen Anforderungen erfasst. Diese dienen nachfolgend in Tabelle 15 als Ausgangslage für einen Blackbox-Test.

Tabelle 15: Blackbox-Test der funktionalen Anforderungen

UC	Titel	Bemerkung	Erfüllt
UC1	Mehrere Bilder importieren		Ja
UC1a	Bilder in der Datenbank persistieren		Ja
UC2	Pro Cluster ein Bild ansehen	Aus zeitlichen Gründen wurde nur ein sehr einfaches Clustering über die ganze Datenbank implementiert. Dieses ist jedoch statisch, d.h. beim Start der Applikation werden immer dieselben Bilder angezeigt.	Nein
UC3	Suchlauf verfeinern (Drilldown)	Ein Drilldown wurde soweit implementiert, dass es möglich ist, jeweils eine neue Suche mit einem oder mehreren Bildern zu starten.	Ja
UC4	Bilder nach Schlagwörtern suchen	Diese Funktionalität wurde mithilfe von Referenzbildern von Google implementiert, d.h. der eingegebene Text in der Suchmaske löst eine Google-Bildersuche aus, deren Resultat dann weiterverarbeitet wird.	Ja
UC5	Bild nach Zutrefflichkeit bewerten	Für den Prototypen wurde im Verlaufe der Studienarbeit entschieden, auf zusätzliches Training zu verzichten.	Nein
UC6	Bild exportieren		Ja
UC7	Installationsskript ausführen	Es wird eine eigenständige Applikation in Form einer .exe-Datei ausgeliefert, die alle Abhängigkeiten bereits enthält.	Ja
UC1234	Suchlauf starten	Ein Suchlauf wird jeweils durch den Benutzer manuell ausgelöst.	Ja

Anmerkung. Eigene Darstellung.

Insgesamt konnten sieben von neun funktionalen Anforderungen umgesetzt werden. Der Use Case UC2 wurde aus zeitlichen Gründen nur ansatzweise realisiert. Da erste Resultate des Inception-v3 Netzes ausreichend waren, wurde der Use Case UC5 nicht länger in Betracht gezogen und es wurde somit gänzlich auf zusätzliches Training verzichtet.

5.3 Test der nicht funktionalen Anforderungen

Zu Beginn dieser Studienarbeit, in Tabelle 4 ersichtlich, wurden nicht funktionale Anforderungen festgelegt. Nachfolgend soll geprüft werden, ob diese erfüllt wurden.

Tabelle 16: Test der nicht funktionalen Anforderungen

Nr.	FURPS+	Titel	Erfüllt
1	Performance	Feature-Berechnung	Ja
2	Supportability	Win 10	Ja
3	+	Datenschutz	Ja
4	Usability	Zusätzliches Training	Nein

Anmerkung. Eigene Darstellung.

Wie aus Tabelle 16 hervorgeht, wurde die funktionale Anforderung Nummer 4 nicht erfüllt. Dies ist darauf zurückzuführen, dass das Inception-v3 Netz ausreichend gute Resultate lieferte und daher auf zusätzliches Training verzichtet wurde.

5.3.1 Profiling der zugrundeliegenden Abläufe

Um den Zeitbedarf der einzelnen Schritte der entwickelten Anwendung auszuwerten, wurden diverse Zeitmessungen durchgeführt. Das verwendete Referenzsystem ist in Anhang G ersichtlich. Tabelle 17 zeigt die Ergebnisse der durchgeführten Zeitmessungen.

Tabelle 17: Zeitmessungen

Operation	Zeit in Sekunden
Berechnung von 15'295 Feature-Vektoren	4045.37
Prüfen und Kopieren von 15’295 Bildern (von HDD nach SSD)	1145.72
Berechnung von 40 Clusterbildern aus 15‘295 Feature-Vektoren	273.05
Suchvorgang ähnlicher Bilder bis zur Anzeige	13.53

Anmerkung. Eigene Darstellung.

Aus den Zeitmessungen geht hervor, dass für das Berechnen der Feature Vektoren am meisten Zeit benötigt wird. Dies war für die Autoren absehbar und wurde in den Anforderungen an die Anwendung berücksichtigt.

Erstaunlicher war jedoch der Zeitbedarf für das Prüfen und Kopieren der Bilder. Bei der Prüfung wird bei jedem Bild einzeln im Hintergrund kontrolliert, ob sich das Bild öffnen lässt. Dies war nötig um sicherzustellen, dass die Bilder valide sind. So werden zum Beispiel einfache Textfiles mit einer Dateiendung wie “.png” gefiltert. Zudem wurden die Bilder aus Gründen des begrenzten Speicherplatzes des Referenzsystems von der langsameren, integrierten HDD auf die SSD kopiert.

Das Berechnen der ersten 40 Clusterbilder dauerte rund 4 Minuten. Das Clustering ist kein zeitkritischer Vorgang, da die Cluster lediglich während eines Bildimports neu berechnet werden. Daher war die Performanz ausreichend.

6 Feedback der Evangelisch-reformierten Kirche Horgen

Nach Fertigstellung des Prototyps konnte dieser erstmals der Evangelisch-reformierten Kirche Horgen präsentiert werden. Die folgenden Punkte wurden rückgemeldet:

1. Die Umsetzung des Prototyps wurde positiv aufgefasst.
2. Die Ergebnisse der Suchen seien brauchbar und sehr interessant.
3. Bei einer Suche sollen als Resultat mehr als nur die besten 40 Resultate angezeigt werden.
4. Die Aktualität der Bilder soll eingegrenzt werden können, wie zum Beispiel nur Bilder aus dem Jahr 2018.
5. Bildduplikate sollen entfernt oder gruppiert werden
6. Zu einem Bild sollen folgende Informationen verfügbar sein:
    a. Autor
    b. Datum
    c. Anlass
7. Nach den im Punkt 4 erwähnten Kriterien soll auch gefiltert werden können.
8. Bei einem Import soll das Datum des Bildes – soweit möglich – automatisch erkannt werden.
9. Eine Liste aller Tags soll zeigen, welche bereits verwendet wurden – diesbezüglich wäre eine Schlagwortwolke denkbar, die auf der Einstiegsseite angezeigt wird.

Die Anregungen wurden als Ausgangslage für die weiterführende Bachelorarbeit aufgenommen.

7 Ergebnisdiskussion

In einem Prototyp konnten die wesentlichen Anforderungen umgesetzt werden. Nachfolgend wird auf die der Studienarbeit zugrundeliegenden Bausteine, Algorithmen sowie Technologien eingegangen und Erkenntnisse erläutert.

Principle Component Analysis (PCA)
Da keinerlei Vorstellung von der Struktur der Daten bestand, wurde mit Hilfe der PCA versucht, eine Dimensionsreduktion vorzunehmen. Ein Experiment mit der Caltech101 Bildersammlung resultierte in 721 Hauptkomponenten, was für eine Clusteranalyse ungeeignet schien. Die Daten liessen sich durch die hohe Anzahl Dimensionen nicht visualisieren.

Manhattan-Distanz und Kosinus-Ähnlichkeit
Die Zeitmessungen der Distanzberechnungen mit einer reinen Python-Implementation zeigte im Gegensatz zur Implementierung von SciPy klare Unterschiede. Die Funktionen der SciPy-Bilbiothek waren auf dem Referenzsystem durchschnittlich dreimal schneller.

In SciPy rechnet die Manhattan-Distanz ca. 19% schneller als die Kosinus-Ähnlichkeit derselben Bibliothek. Bei 10'000 Samples resultierte dies jedoch lediglich in 0.19 Sekunden Zeitdifferenz. Bezogen auf die Geschwindigkeit unterscheiden sich diese beiden Algorithmen somit nur minimal. Vielmehr muss auf die Eignung, bzw. auf den jeweiligen Anwendungsfall geachtet werden.

Anpassungen am Inception-v3 Netzwerk
Das Abschneiden des letzten Layers des Inception-v3 Netzwerkes ergab in der ROC-Kurve eine Verbesserung des AUC-Wertes. Dies war zu erwarten, da die 1000 Klassen des Inception-v3 Netzwerkes nicht auf die Merkmale der Bilder der Evangelisch-reformierten Kirche Horgen passen.

Auswertung der Trefferquote
Während die Applikation bei spezifischen Suchbegriffen lediglich befriedigend abschnitt, zeigte sie dennoch bei einfachen Suchbegriffen wie zum Beispiel “Kind” sehr gute Resultate. Im Gegensatz dazu lieferten abstrakte Suchbegriffe wie “fröhlich” keine brauchbaren Treffer. Das Netzwerk ignoriert Farben oder Helligkeit eines Bildes. Allerdings ist darauf hinzuweisen, dass das Neurale Netzwerk von Google nicht darauf trainiert wurde. Die Resultate einer Suche über mehrere Referenzbilder mittels arithmetischem Mittelwert waren mangelhaft. Somit erweist sich der arithmetische Mittelwert als keine gute Strategie. In einem nächsten Schritt müssten bessere Alternativen evaluiert werden.

Clustering-Verfahren
Da die Anzahl Cluster nicht bekannt war und eine Reduktion der Dimensionen zu keinem Erfolg geführt hat, eigneten sich klassische Clustering-Verfahren wie der k-Means-Algorithmus nicht. Deshalb wurde eigens ein Algorithmus umgesetzt. So lassen sich aus der Bildersammlung eine beliebige Anzahl Bilder extrahieren, die sich von anderen Bildern markant unterscheiden.

Grafische Benutzeroberfläche
Obwohl die Usability nicht im Vordergrund stand, wurde ein relativ einfacher Drag-and-Drop Mechanismus implementiert. Dennoch erwies sich die Navigation und Orientierung bei einer Vorstellung der Software bei der Evangelisch-reformierten Kirche Horgen nicht als intuitiv und verständlich. Auch ist die Applikation nicht attraktiv gestaltet, da Elemente wie Knöpfe nicht ansprechend gestaltet wurden.

Umsetzung in Python
Aus dem Gesichtspunkt eines Softwareentwicklers konnte durch das Einsetzen von SonarCloud und PyLint eine hohe Codequalität sichergestellt werden. Da Python-Programme durch den Global Interpreter Lock (GIL) abgesichert sind, läuft zu jedem Zeitpunkt immer nur ein Thread [36]. Folglich sind Programme mit mehreren Threads langsamer, als wenn die einzelnen Programmteile nacheinander durchlaufen werden. Daher wurden bei zeitkritischen Operationen zusätzliche Prozesse gestartet, die dann auch tatsächlich parallel laufen. Auch wurde die Extrahierung eines Feature-Vektors aus einem Bild auf die GPU ausgelagert, was sich vor allem beim Import von mehreren Bildern in die Datenbank bemerkbar macht.

8 Zusammenfassung und Ausblick

Auf Basis eines siamesischen Netzwerkes konnte ein erster Prototyp umgesetzt werden. Zu einfachen, textbasierten Suchanfragen zeigte dieser sehr gute Treffer an, wohingegen zu abstrakten oder komplexen Suchbegriffen oft keine geeigneten Bilder gefunden werden konnten. Auch ein Suchvorgang mit mehreren Referenzbildern lieferte keine befriedigenden Ergebnisse. Hierbei erweist sich der arithmetische Mittelwert als nicht geeignet. Stattdessen könnten Sparse Matrizen oder eine Singulärwertzerlegung zu einer Verbesserung der Trefferquote und der Berechnungszeit führen.

Die Autoren verwendeten ein eigenes Clustering-Verfahren, welches sich durchaus noch verfeinern lässt. Diesbezüglich wäre eine Anlehnung an die linkage-Implementierung [37] von SciPy, mit dem ein hierarchisches Clustering durchgeführt wird, denkbar.

Die grafische Oberfläche war in erster Linie zu Testzwecken für die Überprüfung der Ergebnisse ausgelegt. In den Bereichen Anwendungsfreundlichkeit, Design und Benutzerführung bietet der Prototyp Verbesserungspotential. Usability Tests und Designanpassungen bieten mögliche Ansätze für Optimierungen.

Letztendlich waren die Autoren überrascht, wie gut sich das Neurale Netzwerk von Google bereits ohne weiteres Training für den Anwendungsfall eignet. In einer Folgearbeit soll der Prototyp die Produktionsreife erreichen und bei der Evangelisch-reformierten Kirche Horgen zum Einsatz kommen.

9 Abkürzungsverzeichnis

AUC	Area under the curve
CNN	Convolutional Neural Network
GIL	Global Interpreter Lock
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
ROC	Receiver-Operating-Characteristic
SSD	Solid State Drive
UC	Use Case

10 Literaturverzeichnis

[1] Szegedy, C, Vanhoucke, V, Ioffe, S, Shlens, J, Wojna, Z (27.06.2016 - 30.06.2016): Rethinking the Inception Architecture for Computer Vision. In: , *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE.

[2] Python Software Foundation (2018): python. https://www.python.org/.

[3] TensorFlow (2018): TensorFlow. https://www.tensorflow.org/.

[4] Raschka, S, Mirjalili, V (04-09-2018): Python machine learning. Machine learning and deep learning with Python, scikit-learn, and TensorFlow. Packt Publishing, Birmingham, Mumbai.

[5] Abadi, M, Barham, P, Chen, J, Chen, Z., Davis, A., Dean, J, Devin, M, Ghemawat, GI, Isard, M, Kudlur, M, Levenberg, J, Monga, R, Moore, S, Murray, DG, Steiner, B, Tucker, P, Vasudevan, V, Warden, P, Wicke, M, Yu, Y, Zheng, X (2016): Proceedings of OSDI '16: 12th USENIX Symposium on Operating Systems Design and Implementation. November 2-4, 2016, Savannah, GA, USA. USENIX Association, Berkeley, CA.

[6] XnSoft (2018): XnSoft. https://www.xnview.com/en/.

[7] Google (2018): Google Cloud Vision API. https://cloud.google.com/vision/docs/detecting-labels.

[8] Egor Chernyshev (2010-2018): Duplicate Finder. http://www.duplicate-finder.com/photo.html.

[9] Slashdot Media (2018): imgSeek. https://sourceforge.net/projects/imgseek/.

[10] Ozone Grif (2013): VisiPics. http://www.visipics.info.

[11] Vitali Fedulov (2018): Similar.Pictures. https://www.similar.pictures/.

[12] Google (2018): Google Fotos. https://www.google.com/photos/about/?hl=de.

[13] Grady, RB (1992): Practical software metrics for project management and process improvement. Prentice Hall, Englewood Cliffs, NJ.

[14] Géron, A (2018): Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow. Konzepte, Tools und Techniken für intelligente Systeme. 1. Auflage. O'Reilly, Heidelberg.

[15] Melekhov, I, Kannala, J, Rahtu, E (04.12.2016 - 08.12.2016): Siamese network features for image matching. In: , *2016 23rd International Conference on Pattern Recognition (ICPR).* IEEE.

[16] L. Fei-Fei, R. Fergus and P. Perona (2004): Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[17] Wikipedia: Manhattan-Metrik. https://de.wikipedia.org/wiki/Manhattan-Metrik.

[18] Wikipedia (2018): Kosinus-Ähnlichkeit. https://de.wikipedia.org/wiki/Kosinus-%C3%84hnlichkeit.

[19] scikit-learn developers (2017-2018): sklearn.metrics.roc_curve. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html.

[20] Scipy community (2014): scipy.spatial.distance.cityblock. https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.distance.cityblock.html.

[21] Scipy community (2018): scipy.spatial.distance.cosine. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html.

[22] Shlens, J (03.04.2014): A Tutorial on Principal Component Analysis.

[23] Richard Jarvis (2018): appJar. http://appjar.info.

[24] Riverbank Computing Limited (2018): PyQt5. https://www.riverbankcomputing.com/software/pyqt/intro.

[25] PySimpleGUI Community (2018): PySimpleGUI. https://pypi.org/project/PySimpleGUI/.

[26] Python Software Foundation (2018): tkinter - Python interface to Tcl/Tk. https://docs.python.org/3.7/library/tkinter.html.
[27] Wieland et al. (2018): Qt History. https://wiki.qt.io/Qt_History.
[28] Google Developers (2018): Google Custom Search. https://developers.google.com/custom-search/.
[29] MIcrosoft (2018): Bing Image Search. https://azure.microsoft.com/en-us/services/cognitive-services/bing-image-search-api/.
[30] Hardik Vasa (2018): google-images-download. https://github.com/hardikvasa/google-images-download.
[31] google-api-python-client Community (2018): google-api-python-client. https://github.com/googleapis/google-api-python-client/tree/master/samples/customsearch.
[32] Wikipedia Community (2018): Hierarchical clustering. https://en.wikipedia.org/wiki/Hierarchical_clustering.
[33] scikit-learn developers (2018): sklearn.metrics.pairwise.cosine_similarity. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html.
[34] Scipy community (2018): numpy.matrix.min. https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.matrix.min.html.
[35] Scipy community (2018): numpy.mean. https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html.
[36] Python Software Foundation (2018): 17.2. multiprocessing — Process-based parallelism. https://docs.python.org/3.6/library/multiprocessing.html.
[37] Scipy community (2018): scipy.cluster.hierarchy.linkage. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html.
[38] MIcrosoft: Microsoft. https://azure.microsoft.com/de-de/services/devops/.
[39] Slack Technologies (2018): Slack. https://slack.com.
[40] PyInstaller Development Team (2018): PyInstaller. https://www.pyinstaller.org.
[41] SonarCloud SA (2018): SonarCloud. https://sonarcloud.io.
[42] PyLint Community (2018): Pylint. https://www.pylint.org.
[43] Guido van Rossum et al. (2001): https://www.python.org/dev/peps/pep-0008/. https://www.python.org/dev/peps/pep-0008/.

Anhang A – Aufgabenstellung

Bildklassifikation mit Hilfe eines neuronalen Netzes

Studienarbeit von: Martin Odermatt und Tobias Saladin

Industriepartner: Reformierte Kirche Horgen

Planung

17.09.2018	Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch den Betreuer.
bis 18.12.2018	Erfassung des Abstracts im Online-Tool https://abstract.hsr.ch/ Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf dem Skripteserver zur Verfügung. Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei.
21.12.2018	Abgabe des Berichts an den Betreuer bis 12.00 Uhr.
21.12.2018	Hochladen der Dokumente auf arachiv-i.hsr bis 12 Uhr.
Regelmässig:	**Wöchentliche Treffen: Mo 15:00 – ca. 16:00 / max. bis 17:00 Uhr** • **Vorbereitung**: Agenda (Themen, Fragen): jeweils bis Fr durch Studierende • **(kurzes) Protokoll**: jeweils bis Dienstag nach dem Meeting

Informationen zur Dokumentation und dem Ablauf der SA insbesondere dem Bericht finden Sie hier:

\\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Studienarbeit_Informatik\SAI14

Der Bericht soll inhaltlich dem „Strukturierungsbeispiel 2" der dort hinterlegten „Anleitung Dokumentation BA/SA" folgen. Der Bericht soll in gedruckter Form ohne Sourcen abgegeben werden. Bei den elektronischen Dokumenten (Sourcen, „Executable", usw.) reicht die Abgabe im Archivserver.

Achtung: Die Bilddatenbank der reformierten Kirche Horgen soll nicht auf den Archivserver geladen werden. Die Dateien unterliegen dem Datenschutz. Bilder, die zur Illustration im Bericht hinterlegt werden, müssen aus Datenschutzgesichtspunkten genehmigt werden (im Zweifel private Fotos oder Fotos des Betreuers verwenden).

Achtung: Die abgegebene Studienarbeit wird mit einer automatischen Plagiatserkennungssoftware überprüft. Bitte unbedingt darauf achten, dass korrekt zitiert wird, d.h. mit Angabe der Quelle. Bei wörtlichen Zitaten sind zusätzlich zur Quellenangabe auch Anführungszeichen erforderlich.

Quellenangaben sind auch erforderlich, wenn es sich um Online-Quellen handelt.

Massives Abschreiben ist keine eigenständige Leistung und führt zur Abwertung der Arbeit.

Aufgabenstellung

Ausgangslage

Die reformierte Kirche Horgen verfügt über eine grössere Bilddatenbank, auf der Fotos von den verschiedensten Anlässen hinterlegt sind. Die Bilder der Datenbank sind gegenwärtig aber ausserordentlich schlecht geordnet und nicht verschlagwortet.

Es ist daher nicht möglich, in der Datenbank gezielt nach Bildern mit bestimmten Charakteristiken zu durchsuchen. Eine Möglichkeit, dieses Problem dadurch zu lösen, die Bilder auf einen Google Account zu laden, und die Bilder mit Hilfe von Google-Funktionalität zugänglich zu machen, wurde aus Datenschutz-Gesichtspunkten wegen dem von Google geforderten „Nutzungsrecht" verworfen.

Das Ziel dieser Arbeit ist es, eine Anwendung zu entwickeln, die die Bilddatenbank für den Endbenutzer zugänglich macht, ohne Datenschutzrechtliche Probleme einzugehen.

Die Idee ist nun eine Anwendung zu entwickeln, die auf Basis von Inception-v3 die vorliegende Bilddatenbank klassifiziert.

Lösungsansatz/Vorgehen

Zunächst besteht die Aufgabe darin, sich in Tensor-Flow und das Inception-v3 zu Grunde liegende Netz einzulesen und eine lauffähige Tensorflow-Umgebung aufzubauen (z.B: mit Hilfe des Tutorials: https://www.tensorflow.org/tutorials/images/image_recognition)

Danach sollen die Bilder der Bilddatenbank mit diesem Netz klassifiziert und geordnet werden. Die Klassifikation soll in diesem Schritt aber nicht (nur) die einem Bild zugeordnete Kategorie speichern, sondern die Features oder Bottlenecks (siehe auch https://www.tensorflow.org/hub/tutorials/image_retraining).

Anhand dieser Feature-Vektoren soll in einem nächsten Schritt ein Browser entwickelt werden, der es erlaubt, zu einem (nicht notwendiger Weise aus der Datenbank stammenden Bild) Bilder mit ähnlichen Features zu finden (Winkel zwischen Feature-Vektoren klein).

Mit Hilfe dieser Funktionalität kann die Bilddatenbank in Cluster aufgeteilt werden, wobei unterschiedliche Cluster aus Bildern und zug. Featurevektoren in unterschiedlichen Ordnern abgespeichert werden.

Falls die Bilddatenbank zu diesem Zeitpunkt bereits gut kategorisiert ist, sind wir an dieser Stelle mit dem Training fertig. Das ist aber nicht zu erwartetn, da die verschiedenen Feature des Featurevektors momentan alle gleich gewichtet sind. In einem nächsten Schritt soll deshalb das bisherige Clustering mit Hilfe eines nachgelagerten neuronalen Netzes „dynamisiert" werden: d.h. das Netz lernt mittels Transfer Learning zunächst die bestehende Clusterung.

Danach soll ein „Cluster-Browser" entwickelt werden, der die Bilder der Cluster so visualisiert, dass die Clusterung manuell leicht korrigiert werden kann. Diese Korrekturen sollen dann durch weiteres Training ins Netz aufgenommen werden.

Schliesslich soll das Benutzerinterface um eine Suchmaske erweitert werden, die dem Benutzer erlaubt, mittels eines drill-downs von wenigen Bildern immer tiefer in die Bilddatenbank einzutauchen.

Der „Bildbrowser" soll ausserdem um eine Funktionalität sinnvoller „Schnittmengenbildung" erweitert werden, so dass man z.B. nach Bildern suchen kann, die den Charakteristiken zweier Cluster entsprechen, oder nach Bildern, die Ähnlichkeit mit zwei vorgegebenen Bildern haben.

Literatur:

Ein gutes Tutorial welches die Entwicklung von AlexNet bis Inception erklärt und die zugrundeliegende Bibliothek tensorflow.slim erklärt

https://cv-tricks.com/tensorflow-tutorial/understanding-alexnet-resnet-squeezenetand-running-on-tensorflow/

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

Publikationen: AlexNet (Deep (!) CNN (Translationsinvarianz)), Inception (GoogLeNet) (Skaleninvarianz?), BN-Inception-v2 (Batch Normalisation)

Inception-v3 Code: git\models\research\inception\inception\slim\inception_model.py

O. Aydin

Anhang B – Persönliche Berichte

Martin Odermatt
Derzeit scheinen Konzepte des maschinellen Lernens geradezu die IT-Welt neu zu erfinden. So ist Facebook zum Beispiel in der Lage, automatisch bei hochgeladenen Bildern jeweils die Gesichter anderer Personen zu identifizieren oder Unternehmen wie Spotify können stark angepasste Musikempfehlungen abgeben. Dabei sind die zugrundeliegenden Konzepte nicht etwa neu, sondern haben in den letzten Jahren durch Themen wie Big Data einen Aufschwung erlebt und ist nun in der Technikwelt in aller Munde.

Eine Studienarbeit in einem vielversprechenden Themengebiet zu schreiben, die sowohl konzeptionelles Verständnis, mathematisches Knowhow als auch Software-Engineering Fertigkeiten abverlangt, machte es interessant aber auch besonders herausfordernd. Da ich vorab noch keine Erfahrungen im Bereich Machine Learning mitbringen konnte und man sich schnell in eine Richtung verlieren kann, war ich um zusätzliche Inputs und Unterstützung seitens des Betreuers - Oliver Augenstein - sehr dankbar.

Die Zusammenarbeit mit meinem Studienkollegen Tobias gestaltete sich nicht nur als sehr angenehm, sondern hat auch viel Spass gemacht und wir konnten neben der komplexen Materie auch mal herzhaft lachen. Ich glaube dies hat auch dazu beigetragen, dass die Motivation durchgehend sehr hoch war.

Schlussendlich bin ich erstaunt, was alles in Kombination mit Open-Source-Bibliotheken und dem frei verfügbaren Inception-v3 Netzwerk möglich ist und freue mich auf eine Folgearbeit.

Tobias Saladin
Ich habe vor dieser Studienarbeit einiges über Machine Learning und Künstliche Intelligenz gehört. Welche Konzepte dahinter stecken war mir jedoch unklar. Ich erhoffte mir durch die Studienarbeit dieses Fragezeichen zumindest teilweise beseitigen zu können. Künstliche Intelligenz ist allgegenwärtig, weshalb es mir ein Anliegen ist, zumindest die Grundkonzepte zu verstehen.

Wir haben mit dieser Studienarbeit nur an der Oberfläche des Machine Learnings gekratzt. Trotzdem konnte ich mir ein Bild über die Vorgehensweisen machen. Die Konzepte ähneln stark dem menschlichen Gehirn. Es fasziniert mich, dass die Technologien grösstenteils auf Basis biologischer Prozesse aufbauen.

Ich kann durchaus sagen, dass mich diese Studienarbeit gefordert hat. Der Einstieg in die Thematik war für mich nicht leicht. Es lohnte sich jedoch sich in die Konzepte einzuarbeiten und die Lösungsansätze zu verstehen. Ich entdeckte damit die scheinbar unendlichen Möglichkeiten wie auch die Grenzen von Machine Learning. Da sowohl Martin als auch ich keine Erfahrungen auf diesem Gebiet hatten, war die Zeit knapp, um nebst der Einarbeitungszeit auch ein fertiges Produkt herzustellen. Das es zeitlich trotzdem gereicht hat ist vor allem der Teamarbeit zu verdanken. Martin und ich haben uns sehr gut in unseren Fähigkeiten ergänzt.

Gerne hätte ich weiter die Resultate durch mathematische Optimierungen verbessert und so neue Lösungsansätze kennengelernt. Aus diesem Grund freue ich mich besonders, dass durch die Begleitperson vorgeschlagen wurde, das Projekt während der Bachelorarbeit weiter zu vertiefen.

Anhang C – Zeitabrechnung

Der Aufwand der Arbeitspackete wurde anhand von Storypoints gewichtet. Zehn Storypoints entsprechen rund acht Stunden Arbeit. In den Tabellen 18 bis 23 sind sämtliche Stories nach Sprints gegliedert:

Tabelle 18: Sprint 1, 23.09.2018 – 07.10.2018

Stories	T. Saladin	M. Odermatt
Weekly Team Meeting KW38		1
Bibtex aufsetzen		3
Disposition		5
Latex Einrichtung		5
CI für Latex inkl. Biber		6
TensorFlow: Cut layer		8
Über Tensor Board informieren	2	
Risiken erfassen	2	
Latex kennenlernen	3	
Informationen sammeln für 1. Tutorial	5	
Erster Entwurf Aufgabenstellung und Technologien in SA	5	
Aufsetzen des Grundgerüsts von TensorFlow mit Tutorial	12	
Meeting mit Oliver KW39	2	2
Meeting mit Oliver KW38	3	3

Anmerkung. Eigene Darstellung.

Tabelle 19: Sprint 20, 07.10.2018 – 21.10.2018

Stories	T. Saladin	M. Odermatt
Sitzungsprotokoll KW39		1
E-Mail an Oliver KW 40		1
Email an Oliver KW41		1
Evaluation Glossar für Latex		3
FA + NFA überarbeiten		5
Use Cases überarbeiten		5
Storyboarding -> Strukturierung des GUI, Wireframes		10
Principle Component Analysis		15
Besprechung mit Nadine wegen Darstellung der Doku	1	
Estimated Story Points nachtragen	1	
Export Zeiterfassung + Dok	1	
Weekly Scrum Meeting WK40	3	
Evaluation GUI	5	
Algorithmen für Bildvergleich vergleichen mit Genauigkeit und Performance	5	
REST Google oder Microsoft	5	
Umsetzung ähnliche Bilder finden	5	
Umsetzung der Inputs von Nadine	8	

Weekly Team Meeting KW39	1	1
Meeting mit Oliver KW40	3	3

Anmerkung. Eigene Darstellung.

Tabelle 20: Sprint 3, 21.10.2018 – 04.11.2018

Stories	**T. Saladin**	**M. Odermatt**
Doku, Meeting aktualisieren KW 43		1
Installation QT Designer		4
PCA überarbeiten		4
Tools		5
PyQT5 kennen lernen	2	5
PCA dokumentieren		5
GUI -> funktionierender Prototyp		35
Weekly Meeting KW42	4	4
Meeting mit Oliver KW41	1	
Dokumentation ROC Kurve	2	
Doku updaten(Reviews, Timetable)	2	
Bilder der Kirche vektorisieren	3	
Automatisierte ROC Kurve	5	
Network Graph plotten	5	
Softmax mit Manhattan	6	
Receiver-Operator-Characteristic-Kurve	13	
Scrum Meeting KW 43	3	3

Anmerkung. Eigene Darstellung.

Tabelle 21: Sprint 4, 04.11.2018 – 18.11.2018

Stories	**T. Saladin**	**M. Odermatt**
Bugfix: Folder "Suchresultate" muss erstellt werden		1
Email an Oliver: Update KW45		1
Bilder Importieren GUI Umsetzung		12
continuous integration python		12
3-Level suche mit gutem Default		
Google Suche unabhängig von Resultat Suche	1	
Reihenfolge der Resultatbilder überprüfen	1	
Bilder einzeln löschen	2	
loadtxt tuning	2	
Usability für Löschen der Bilder in Stacklist optimieren	4	
Bilder nach Trefferquote sortieren	4	
Ordnerstruktur anpassen	5	
Meeting mit Oliver KW 43	3	3
Meeting mit Oliver KW44	2	2

Anmerkung. Eigene Darstellung.

Tabelle 22: Sprint 5, 18.11.2018 – 02.12.2018

Stories	T. Saladin	M. Odermatt
Bugfix: File exists: 'Suchresultate/Suche_101/Resultat'		1
Bugfix: support png etc.		1
Refactor in Image service, create_new_search_folders		1
check pathlib case sensitivity		3
Bild speichern unter		4
SonarLint		4
PyLint		6
Hinweis wenn keine Bilder DB ausgewählt		11
Bugfixing	4	4
Refactor duplicated Code	1	
Processes for Neighbors Thread	1	
Namensgebung der Files anpassen	2	
Add tests	3	
Processes for Index Thread	4	
Evaluate Processes for Images Thread	4	
Clustering mit ROC	5	
Optimize Feed Cluster	8	
Meeting mit Oliver KW44	2	2
Meeting mit Oliver KW45	2	2
Clustering	17	

Anmerkung. Eigene Darstellung.

Tabelle 23: Sprint 6, 02.12.2018 – 21.12.2018

Stories	T. Saladin	M. Odermatt
Dokumentation	100	100
Meeting Evangelisch-reformierte Kirche Horgen	5	5

Anmerkung. Eigene Darstellung.

Anhang D – Protokolle der Sitzungen

In den Tabellen 24 bis 35 sind die Angaben zu sämtlichen Sitzungen protokolliert:

Tabelle 24: Sitzungesprotokoll KW38

Woche	KW38
Datum & Zeit	17.09.2018 / 15:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Kickoff • Übergabe der Aufgabenstellung und Erläuterungen
Beschlüsse	• Es findet feweils ein wöchentliches Treffen statt • Der Fokus in dieser Woche soll auf das kennenlernen von TensorFlow gerichtet werden

Anmerkung. Eigene Darstellung.

Tabelle 25: Sitzungesprotokoll KW39

Woche	KW39
Datum & Zeit	25.09.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Inhalt & Umfang der Arbeit • Anforderungsanalyse
Beschlüsse	• Umfang kürzen und mehr in die Einleitung verschieben • Theorieteil weglasssen • Fokus auf ”Cut last layer“ legen • Einleitung mit Ausgangslage fehlt • Inception erläutern

Anmerkung. Eigene Darstellung.

Tabelle 26: Sitzungesprotokoll KW40

Woche	KW40
Datum & Zeit	02.10.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Kapitel Research • Clustering Verfahren • Weiteres Vorgehen
Beschlüsse	• Über PCA informieren • Lösung für das Finden von ähnlichen Bildern evaluieren

Anmerkung. Eigene Darstellung.

Tabelle 27: Sitzungesprotokoll KW41

Woche	KW41
Datum & Zeit	09.10.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Google Suche • Clustering
Beschlüsse	• Lösung für die Suche mit Google API umsetzen • Clustering Verfahren evaluieren, bestehende Libraries können benutzt werden

Anmerkung. Eigene Darstellung.

Tabelle 28: Sitzungesprotokoll KW42

Woche	KW42
Datum & Zeit	16.10.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Bilder der Kirche Horgen • Neue Lösung für Ähnlichkeit von Bildern mittels Manhattan Algorithmus
Beschlüsse	• Umsetzung mit Manhattan Algorithmus prüfen • Auf die Bilder der Kirche Horgen kann nun online zugegriffen werden

Anmerkung. Eigene Darstellung.

Tabelle 29: Sitzungesprotokoll KW43

Woche	KW43
Datum & Zeit	23.10.2018 / 07:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• ROC-Kurve für Verifikation plotten
Beschlüsse	• Vektoren aus allen Bildern berechnen • Matrix berechnen • Verstehen, was die Bibliothek macht

Anmerkung. Eigene Darstellung.

Tabelle 30: Sitzungesprotokoll KW44

Woche	KW44
Datum & Zeit	30.10.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• ROC-Kurve mit true & false positives • Graph
Beschlüsse	• statt visualisieren, mehr Metriken berechnen • Testen, dass Manhattan funktioniert oder eben nicht • Bilder im Bericht auslassen, da problematisch • Manuelle Tests erwünscht

Anmerkung. Eigene Darstellung.

Tabelle 31: Sitzungesprotokoll KW45

Woche	KW45
Datum & Zeit	6.11.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Bing oder Google? • Performance
Beschlüsse	• Google-Bildsuche als Referenz nehmen • Optimierung wichtig • Summe der Diagonalelemente einer Matrix berechnen • Singular Value Decomposition anschauen

Anmerkung. Eigene Darstellung.

Tabelle 32: Sitzungesprotokoll KW46

Woche	KW46
Datum & Zeit	13.11.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Ledoit-Wolf • Multi-Gauss • Besprechung BA
Beschlüsse	• Assume_centered wie default belassen • Ledoit-Wolf braucht mindestens 3 Matrizen • Mitteilen, ob eine Folgearbeit als BA gewünscht ist

Anmerkung. Eigene Darstellung.

Tabelle 33: Sitzungesprotokoll KW48

Woche	KW48
Datum & Zeit	27.11.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Bilderfreigabe • Gemeinsamer Termin mit der Reformierten Kirche
Beschlüsse	• Performance-Tuning wichtig für diese Studienarbeit • Workflow durchspielen mit der Reformierten Kirche • Outcome im Bericht belegen mit Beispielen

Anmerkung. Eigene Darstellung.

Tabelle 34: Sitzungesprotokoll KW49

Woche	KW49
Datum & Zeit	7.12.2018 / 14:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin, Petra Gassmann, Tiana
Traktanden	• Vorstellung des Prototyps • Typischer Anwendungsfall “Flyer” durchspielen • Anforderungen und Feedback aufnehmen • Publizieren von Bildern
Beschlüsse	• Anlass, Autor, Datum beim importieren • Mehr Suchresultate darstellen • Tagcloud oder ähnliches wäre schön • Bildduplikate entfernen oder gleiche Bilder gruppieren • Feature Bildähnlichkeit zu Tagähnlichkeit denkbar • Filter nach Aktualität (z.B. nur Bilder von 2018) wünschenswert • Filter von Bildern nach Autor (z.B. nur Tiana)

Anmerkung. Eigene Darstellung.

Tabelle 35: Sitzungesprotokoll KW50

Woche	KW50
Datum & Zeit	11.12.2018 / 17:00 Uhr
Teilnehmer	Oliver Augenstein, Martin Odermatt, Tobias Saladin
Traktanden	• Inhalt der Studienarbeit
Beschlüsse	• Cosine-Metrik geschickter schreiben • Bildvergleiche mehr interpretieren • Beschreibung des Siamesischen Netzwerks wäre gut

Anmerkung. Eigene Darstellung.

Anhang E – Infrastruktur und Tools

Nachfolgend sind die verwendeten Tools aufgelistet.

Azure DevOps

Azure DevOps [38] (früher Visual Studio Team Services) ist eine Plattform, die Entwicklern agile Tools zur Verfügung stellt. Für dieses Projekt werden das grafische Board, die Git-Repositories und das Continuous Integration von Azure DevOps verwendet.

Board

Damit die anstehenden Aufgaben besser geplant und verfolgt werden können, eignet sich ein wie in Abbildung 19 dargestelltes, grafisches Board.

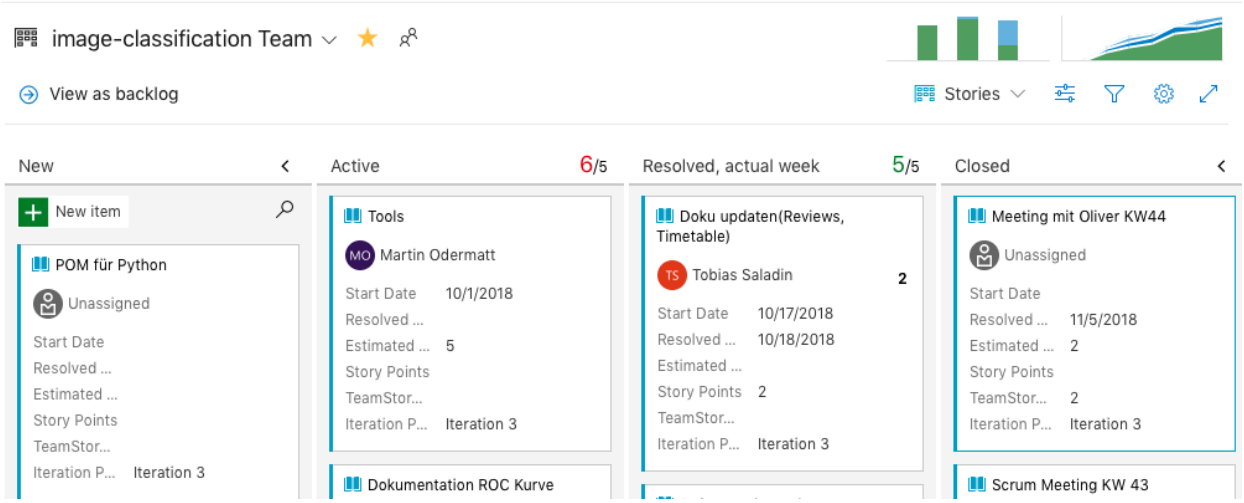

Abbildung 19: Kanban-Board aus Azure DevOps

Anmerkung. Eigene Darstellung.

Das Projekt ist in zwei Repositories gegliedert:

- Prototypes: Eine Art Research-Umgebung
- Image-classification: Der eigentliche Source Code für die auszuliefernde Software

Continuous Integration

Bei jedem Speichern ins Repository wird ein Build-Prozess angestossen. So kann sichergestellt werden, dass die Software stets lauffähig gehalten wird und Fehler frühzeitig erkannt werden.

Slack

Damit die Zusammenarbeit im Team optimal funktioniert, ist ein guter Kommunikationskanal unabdingbar. Slack [39] ist eine Plattform für die kollaborative Zusammenarbeit. Neben dem Austausch von einfachen Nachrichten können auch Dienste von Drittanbietern angehängt werden. So wird wie in Abbildung 20 ersichtlich ist, ein Kanal genutzt, um über die neusten Build-Resultate zu informieren.

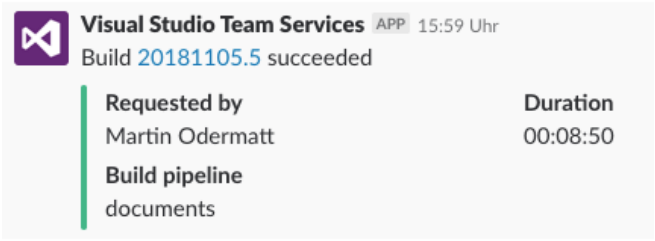

Abbildung 20: Azure DevOps Meldung an Slack

Anmerkung. Eigene Darstellung.

PyInstaller

PyInstaller [40] ist ein Programm, mit der sich andere Python-Programme in eigenständige, ausführbare Dateien einfrieren lassen. So braucht das Zielsystem keine Module oder Pakete und nicht einmal Python selbst, installiert zu haben. Das fertige Artefakt kann somit sehr einfach auf ein neues System kopiert und gleich in Betrieb genommen werden. Dies wird für die Installation bei der Evangelisch-reformierten Kirche Horgen verwendet.

Softwarequalität

Um die Code-Qualität hoch zu halten, wurde vom Continuous Integration System nach jedem Build eine Codeanalyse mittels SonarCloud [41] angestossen. SonarCloud unterstützt über 20 Programmiersprachen und erkennt die gängigsten Code Smells, Bugs und Vulnerabilities. Dabei werden die gefundenen Resultate in einer grafischen Benutzeroberfläche aufbereitet, wie sie in Abbildung 21 zu sehen ist.

Abbildung 21: SonarCloud

Anmerkung. Eigene Darstellung

Wie Abbildung 22 zeigt, wird zu einer vordefinierten Regel jeweils eine kurze Beschreibung angezeigt.

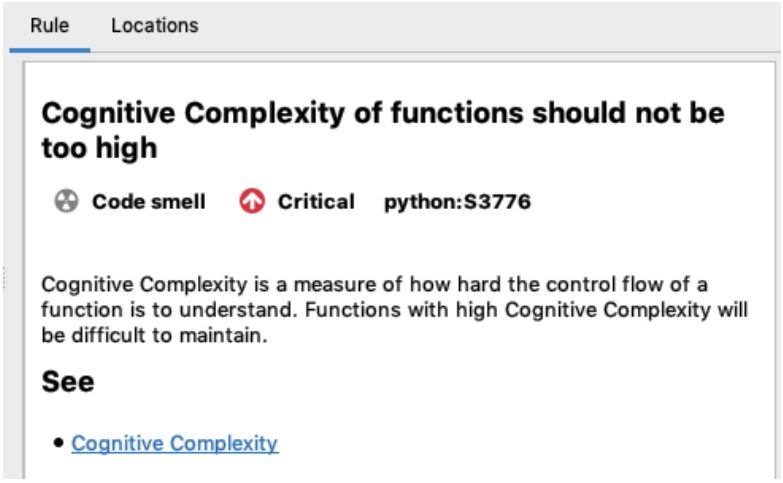

Abbildung 22: SonarCloud Regeln

Anmerkung. Eigene Darstellung.

In diesem Fall handelt es sich um eine Verletzung der “Cognitive Complexity Rule”, die besagt, dass eine Funktion nicht länger als 15 Zeilen Code sein darf, da sie sonst nur schwer wartbar

wird. Für die komfortable Handhabung wurde auf den Entwicklungsumgebungen der Autoren ein Plugin namens “PyLint” [42] installiert, womit die Analyse von SonarCloud jeweils angezeigt wird. So ist es zum Beispiel möglich, direkt zur jeweiligen Code-Zeile zu springen. Ausserdem wird für die einheitliche Codeformatierung ein Makro installiert, dass den Source Code automatisch nach dem Pep8-Standard [43] formatiert.

Anhang F – Beispiel Resultate bei Suche nach ähnlichen Bildern

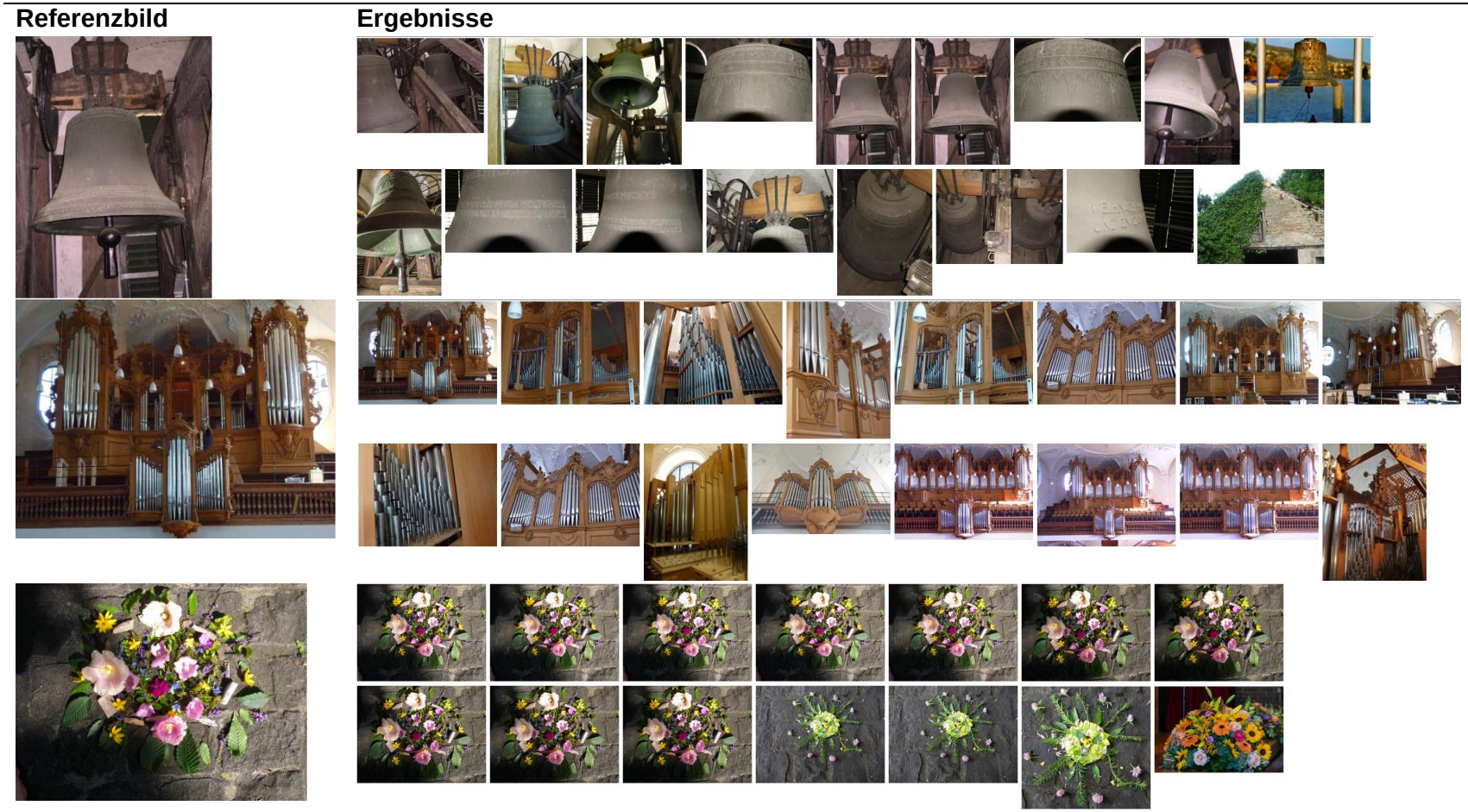

Abbildung 23: Veranschaulichung der ähnlichen Bilder Suche

Anmerkung. Eigene Darstellung.

Anhang G – Referenzsystem

Für die gesamte Studienarbeit wurde auf nachfolgendem Referenzsystem gearbeitet. Sämtliche Zeitmessungen wurden mit den angegebenen Systemkomponeten durchgeführt.

- CPU: Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz
- GPU: GeForce GTX 960M
- RAM: 16 GB DDR3
- SSD: Kingston RBU-SNS8100S3256GD
- HDD: WDC WC10JPVX-22JC3T0