

**FOM Hochschule für Ökonomie und Management**

**Standort Köln**

**Berufsbegleitender Studiengang zum**

**Master of Science Big Data & Business Analytics**

**2. Semester**

**Projektarbeit im Modul Analyse semi- und unstrukturierter Daten**

**Bilderkennung von Tieren mit Cloud Services**

Betreuer : Professor Dr. Arthur Dill

Autoren : Neli Garkova (528396)

Dominik Mazurkiewicz (524895)

Simon Schell (518607)

Abgabedatum : 31.08.2020

## Inhaltsverzeichnis

I	Abbildungsverzeichnis .....	IV
II	Abkürzungsverzeichnis .....	V
1	Motivation .....	1
1.1	Einleitung.....	1
1.2	Projekt .....	3
1.2.1	Projektbegründung .....	3
1.2.2	Projektziele .....	4
1.2.3	Projektplanung.....	4
2	Theoretischer Hintergrund und verwandte Arbeit.....	7
2.1	Theoretischer Hintergrund.....	7
2.2	Verwandte Arbeiten .....	10
3	Datensatz.....	13
3.1	Problemstellung .....	13
3.2	Webcrawling .....	13
3.3	Wahl des Werkzeugs .....	14
3.4	Erstellung der Crawler.....	14
3.5	Bildmodifikation .....	17
4	Methodik .....	18
4.1	Experimentelles Setup.....	20
4.2	Erkennung unter herausfordernden Bedingungen .....	23
4.2.1	Erstellung von Adversal Examples.....	24
5	Ergebnisse.....	24
5.1	Ergebnisse Phase 1 .....	25
5.2	Ergebnisse Phase 2 .....	25
5.3	Ergebnisse Phase 3 .....	27

6	Diskussion .....	29
7	Fazit .....	30
	Literaturverzeichnis .....	32
	Ehrenwörtliche Erklärung .....	36

## I Abbildungsverzeichnis

Abbildung 1: "Projektablaufmodell" in Anlehnung an: Buchkremer et al. (2019), S. 4 ... 6	
Abbildung 2: Übersicht der Eigenschaften der jeweiligen Phasen .....	20
Abbildung 3: Vergleich der verschiedenen Cloud Anbieter .....	23
Abbildung 4: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 1 .....	25
Abbildung 5: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 2 (Rauschen) ....	26
Abbildung 6: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 2 (Unschärfe) ...	26
Abbildung 7: Erkennungsgenauigkeit der Cloud Anbieter nach Bildmodifikationen .....	27
Abbildung 8: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 3 (Rauschen) ....	28
Abbildung 9: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 3 (Unschärfe) ...	28

## II Abkürzungsverzeichnis

ANN	Artificial Neural Network
API	Application Programming Interface
AWS	Amazon Web Services
BRIEF	Binary Robust Independent Elementary Features
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Networks
GCP	Google Cloud Platform
ML	Machine Learning
SIFT	Scale-invariant feature transform
SURF	Speeded up robust features

# 1 Motivation

## 1.1 Einleitung

Computer Vision zählt zu den wichtigsten Wachstumsbereichen der künstlichen Intelligenz. Laut Forbes soll der Markt von 9,9 Milliarden US-Dollar in 2019 auf 14,4 Milliarden US-Dollar in 2024 wachsen<sup>1</sup>. Computer Vision ist eine Disziplin, die Computern die Interpretation und das Verständnis der visuellen Welt ermöglicht. Die automatisierte Bildanalyse ist Teil der Computer Vision, die in vielen Bereichen verwendet wird. Der anspruchsvollste Bereich in der Bildanalytik ist die Objektklassifizierung. Sie benötigt einen Algorithmus, der verlässlich alle Merkmale eines Objekts wiedererkennt, und zwar unabhängig vom Bildhintergrund. Durch Fortschritte im Bereich von Deep Learning sind beeindruckende Ergebnisse hinsichtlich der Genauigkeit und Geschwindigkeit bei der Bilderkennung möglich geworden.

Der Erfolg von Deep Learning-Ansätzen für Computer Vision hat zu einer Explosion der Nachfrage geführt. Um die Verwendung von DL-Algorithmen zu erweitern und zu vereinfachen, haben Cloud Anbieter wie Amazon, Google, Microsoft, IBM und andere Computer Vision-bezogene Dienste entwickelt, mit denen Nutzer auch ohne Fachkenntnisse komplexe neuronale Netze schnell entwerfen, entwickeln und trainieren können. Dabei entfällt der Mehraufwand durch die Verwaltung eigener Hardware. Cloud Provider bieten außerdem zusätzliche Unterstützung an, wie z.B. Vision-APIs, mit deren Hilfe sich Bilder klassifizieren und in natürlicher Sprache beschreiben lassen.

Trotz ihrer beeindruckenden Leistung haben Forschungen gezeigt, dass unterschiedliche Möglichkeiten existieren, um DNN Modelle gezielt zu täuschen. Damit kann versucht werden, das Netzwerk zu einer Falschklassifizierung zu bringen. Sehr beeindruckende Ergebnisse lassen sich mit sogenannten Adversarial Examples erzielen<sup>2</sup>. Dabei werden realistisch aussehende Bilder minimal verändert. Das menschliche Auge nimmt den Unterschied praktisch nicht wahr. Dennoch identifiziert das DNN im manipulierten Bild

---

<sup>1</sup> Vgl. Marr (2020), S. 1

<sup>2</sup> Vgl. Dodge et al. (2016), S. 4ff.

einen völlig anderen Gegenstand. Die Forscher hielten diese Störversuche zunächst für sehr theoretisch, bis eine Gruppe von Studenten der MIT-Organisation LabSix zeigte, dass sie dreidimensionale Objekte erstellen können, die DL-Algorithmen zur Falschklassifizierung bringen. Dies zeigte, dass Adversarial Examples in der realen Welt ein Problem darstellen<sup>3</sup>, robuste Computer-Vision-Modelle zu entwerfen.

Die Sicherheitsbewertung von ML-Systemen ist ein aufstrebendes Forschungsgebiet. In mehreren Veröffentlichungen wurden Adversarial Angriffe auf verschiedenen DL-Algorithmen für Computer Vision vorgestellt. Während einige Untersuchungen ausführlich gezeigt haben, dass White-Box Systeme für Adversarial Examples anfällig sind, erhalten Cloud-basierte Bilderkennungsmodelle, die ähnliche Sicherheitsbedenken haben allerdings als Black-Box Systeme betrachtet werden können, bisher noch wenig Aufmerksamkeit. Um diese Lücke zu schließen, werden in dieser Arbeit die Bildklassifizierungsdienste der drei wichtigsten Cloud-Plattformen betrachtet: Amazon Web Services (AWS), Google Cloud Platform (GCP) und Azure. Das Ziel der Forschung ist es, die Services anhand ihrer Performance und Handhabung zu vergleichen und die Robustheit der Klassifizierer gegenüber Adversarial Examples zu untersuchen. Im Rahmen dieser Forschungsarbeit soll geprüft werden, ob die oben genannten Services sich durch Bildmodifikationen leicht täuschen lassen. Dabei wird untersucht, wie sich die Erkennungsleistung der Algorithmen unter anspruchsvollen Bedingungen wie Bildrauschen und Unschärfe ändert. Für die Untersuchungen wurde ein Datensatz mit Bildern von verschiedenen Tieren genutzt, weil diese Bilder im Internet einfach und in ausreichender Anzahl zu finden sind und durch Menschen leicht klassifiziert werden können.

Die vorliegende Arbeit ist folgendermaßen aufgebaut: Zunächst werden innerhalb dieses Kapitels die übergeordneten Ziele und Begründungen des Projekts geschildert. Das zweite Kapitel beschreibt den theoretischen Rahmen und die relevante Literatur. In Kapitel 3 werden die Techniken und Verfahren zur Datenbeschaffung aus den Quellsystemen behandelt. Anschließend wird in Kapitel 4 die Methodik der

---

<sup>3</sup> Vgl. "Fooling Google's image-recognition AI 1000x faster " (2017)

Untersuchung dargestellt. Im 5. Kapitel der Arbeit werden die Forschungsergebnisse präsentiert, nachfolgend in 6. diskutiert, gefolgt von dem Fazit in Kapitel 7.

## 1.2 Projekt

### 1.2.1 Projektbegründung

Zur Anwendung von Computer Vision bzw. der in dieser Arbeit behandelte Bereich der Image Classification gibt es eine große Bandbreite an Use Cases. Diese reichen beispielsweise von Implementierungen innerhalb von Unternehmen, z.B. durch die Automatisierung der Landwirtschaft<sup>4</sup>, über soziale Fallbeispiele bis hin in den Bereich der Umwelt. Aufgrund der Verschärfung der Folgen des Klimawandels wurde sich für ein solches Szenario entschieden. Eine dieser Folgen ist laut Nature, dass eine Millionen Arten vor dem Aussterben stehen<sup>5 6</sup>. Die Bildanalyse von Tieren ist ein Teil eines größeren Konstrukts, welches Tier- und Artenschutzorganisationen hilfreiche Informationen liefern können, um die Folgen abzuschwächen. Wird nur der konstruierte Fall betrachtet, dass Bildern von Wildtierkameras (damit sind in der Natur aufgestellte Kameras mit automatischer Auslösung bei Bewegung gemeint) durch das trainierte Modell Labels zugewiesen werden können, sind diese Informationen für die Organisationen z.B. die Wiedergabe der Biodiversität. Das bedeutet Wissen darüber, wie viele verschiedene Arten es in einem bestimmten Gebiet gibt und eine Einschätzung darüber, wie groß Populationen sind. Weiterhin können Tiere als Indikator dienen, um die Qualität eines Habitats wieder zu geben. Die gelabelten Bilder, der daran angeknüpfte Zeitstempel der Auslösung der Kamera und die Geolocation des Ortes der Kamera sind dafür die zugrundeliegenden Daten.

Über den konstruierten Fall hinaus geht, dass es mit wenig zusätzlichen Aufwand möglich ist, weitere Daten zu erheben. Das sind beispielsweise Sensordaten, die die Luft-/Bodenfeuchtigkeit, Temperatur, Windgeschwindigkeit, Niederschlag etc. wiedergeben

---

<sup>4</sup> Vgl. Tian et al. (2020), S. 15

<sup>5</sup> Vgl. Tollefson (2019), S. 171

<sup>6</sup> Vgl. Román-Palacios et al. (2020), S. 3



können. Damit würden den Organisationen weitere Daten vorliegen, aus denen diese noch mehr Informationen generieren können.

Ein Beispiel: Zebras werden in der Savanne nicht mehr so oft an einer Fotofalle aufgenommen und klassifiziert, wie die Jahre zuvor. Außerdem hat die Temperatur zugenommen und die Bodenfeuchtigkeit abgenommen. Aus diesen Informationen könnte geschlossen werden, dass das ursprüngliche Habitat nicht mehr attraktiv für die Zebras ist aufgrund einer fortschreitenden Desertifikation.

### 1.2.2 Projektziele

Die Projektziele lassen sich aus den folgenden Fragen ableiten:

1. Kann mit den knappen Ressourcen Budget (bei den Cloud-Services) und Zeit ein Modell trainiert werden, was sinnvoll in dem Praxiskontext Anwendung finden kann? Das Ziel ist somit unter den gegebenen Limitationen ein End-to-End Modelltraining durchzuführen mit möglichst hoher Genauigkeit.
2. Welcher der Cloud-Services weist die beste Umsetzung auf? Das Ziel ist dem am besten für den Anwendungsfall geeigneten Cloud-Service zu finden.
3. Wie beeinflussen Adversarial Examples (durch Bildmodifikationen erzeugt) die Performance? Das Ziel ist eine Validierung der Papers mit gleichen Untersuchungshintergrund auf dem gewählten Anwendungsfall durchzuführen.

### 1.2.3 Projektplanung

Der Projektplanung war das Vorgehensmodell der STIRL Doppeltrichters für künstliche Intelligenz zu Grunde gelegt<sup>7</sup>. Für den Anwendungsfall bei Computer Vision wurde das Modell angepasst, indem nicht benötigte Phasen weggelassen worden sind bzw. neue wichtige hinzugefügt werden mussten. Die Entwicklung des Ablaufmodells war zum Projektstart die wichtigste Aktion, um eine Vorstellung der wichtigsten zu benötigten Komponenten zu bekommen. Zunächst wurde eine sehr simple End-to-End

---

<sup>7</sup> Vgl. Buchkremer et al. (2019), S. 3ff.

Minimallösung entwickelt, eine Art von Proof-of-Konzept. Diese war lediglich in Datenbeschaffung, Integration in den AWS Cloud-Storage, das Trainieren des AWS Image Rekognition Modells sowie das Testing ausgelegt. Bei diesen geringen Datenmengen haben sich Abweichungen zu den späteren umfangreichen Modelltrainings ergeben. Beispielsweise waren sehr wenige Bilder von weiblichen Löwen vorhanden und aufgrund des unterschiedlichen Phänotyps resultierte daraus eine schlechte Labeling-Performance im Vergleich zum männlichen Löwen. Intra-Class-Variation und ein Übergewicht an Trainingsdaten einer daraus resultierenden Ausprägung sind bekannte Herausforderung im Bereich der Image-Classification.

Trotzdem war bis auf genannte Ausnahme die Performance zufriedenstellend, sodass die Weiterentwicklung sinnvoll war. Zum einen wurde die Entscheidung getroffen, nicht nur den Cloud-Service AWS, sondern auch Azure und Google Cloud Platform zu nutzen. Damit soll eine Evaluation der verschiedenen Cloud-Services ermöglicht werden, indem die Modelle unter gleichen Bedingungen trainiert werden. Weiterhin ist durch die Datenbeschaffung von den gewählten Bildplattformen, von denen die Bilder unter anderem beschafft wurden, nur qualitativ hochwertiges Bildmaterial im Datenkorpus verfügbar. Mit diesem Hintergrund und weitere Recherchearbeit konnte das Ablaufmodell durch optionale Bildmodifikationen erweitert werden. Diese beschäftigen sich mit für die Genauigkeit relevanten Qualitätskriterien von Bildern wie Bildrauschen oder Unschärfe.

Weiterhin mussten auch notwendige Bildmodifikationen mit in den Ablauf genommen werden, weil das Training in den Cloud-Services diese erforderten. Diese weiteren Punkte sind allerdings erst nach Projektstart hinzugekommen, weil erst in einer späteren Iteration auf einer größeren Trainingsdatenmenge diese Faktoren aufgefallen sind. Damit hat das Ablaufmodell sich im Laufe der Zeit von einer Minimallösung zu einem vielfältigen Modell entwickelt.

Soweit es möglich war, wurden einzelne Teile abstrahiert und entkoppelt. Das war sehr gut mit der Datenbeschaffung möglich, nicht aber zwischen Cloud-Storage und Cloud-Service. Bei Letzterem war die Abhängigkeit gegeben, dass eine ausreichende Anzahl von Bildern im Storage vorliegen musste, um den CV-Service darauf zu starten. Damit das Training überhaupt gestartet werden konnte musste eine Mindestanzahl an Bildern

im Storage vorliegen. Weiterhin war das Training der Modelle in den Cloud Services an sich voneinander unabhängig. Diese Entkopplung ermöglichte es viele Schritte zu parallelisieren und in der gleichen Zeit mehr Fortschritt zu erzielen. Allerdings musste dabei immer beachtet werden, nicht das Zusammenfügen zu einem späteren Zeitpunkt zu gefährden. Dem Projektteam war es somit möglich, sich auf spezifische Problemdomänen des Gesamtkonstrukts zu spezialisieren und diese mit mehr Expertise zu behandeln.

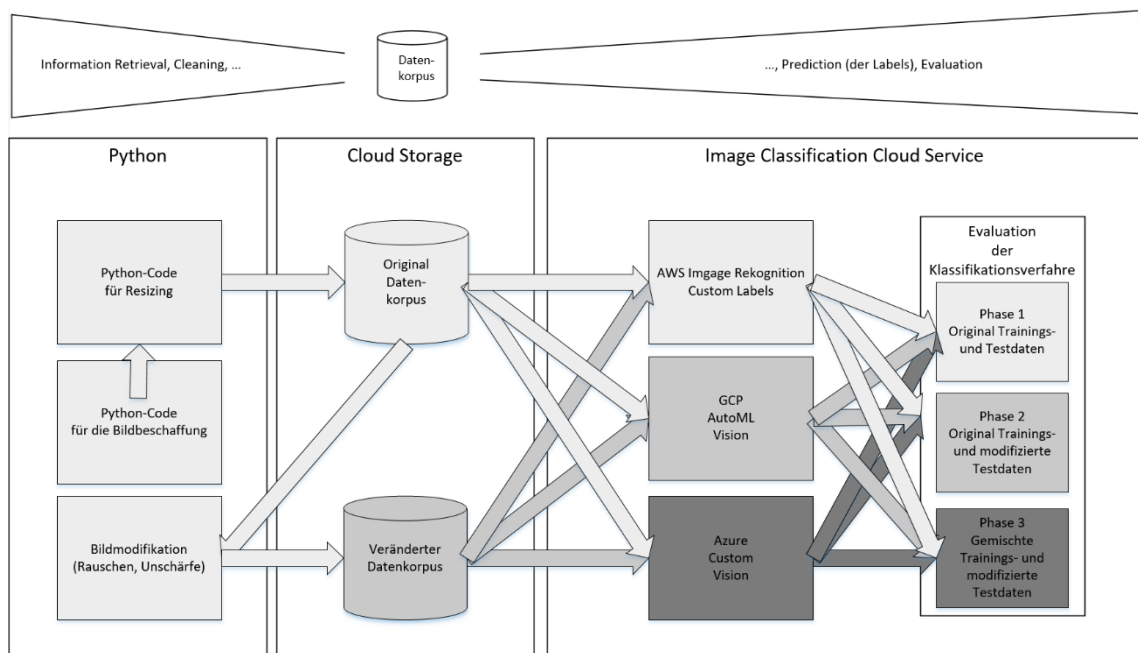


Abbildung 1: "Projektablaufmodell" in Anlehnung an: Buchkremer et al. (2019), S. 4

Bei der Abbildung wurde aus Gründen der Übersichtlichkeit darauf verzichtet mit weiteren Pfeilen die zahlreichen Iterationen darzustellen. Prinzipiell wurde bei allen Schritten mit einem einfachen Proof-of-Concept gestartet und dieser bei Erfolg erweitert. Beispielsweise wurden die Modelle erst mit 30, dann mit 300 und zum finalen Zeitpunkt mit 1000 Bildern durchlaufen und evaluiert. Somit konnten Limitationen und weitere Herausforderungen schneller aufgedeckt werden, damit diese so früh wie möglich beseitigt werden konnten.

## 2 Theoretischer Hintergrund und verwandte Arbeit

### 2.1 Theoretischer Hintergrund

CV ist ein multidisziplinäres Forschungsgebiet, das sich damit beschäftigt, Techniken und Methoden zu entwickeln, mit denen die Fähigkeit des menschlichen Sehens reproduziert wird. Mit der Unterstützung von CV sollen Computer digitale Bilder auf die gleiche Weise wie Menschen wahrnehmen und verstehen können.<sup>8</sup> Im Bereich CV versuchen Forscher, Computer dazu zu bringen, die Welt um sie herum zu „sehen“, zu verstehen und dann entsprechend zu handeln. Dies wird durch eine Kombination von Hardware und Software erreicht. Ziel ist es:

- "die Konstruktion von aussagekräftigen Beschreibungen physischer Objekte aus Bildern" (Ballard & Brown, 1982)
- "Berechnung von Eigenschaften der 3D-Welt aus einem oder mehreren digitalen Bildern" (Trucco & Verri, 1998)
- "nützliche Entscheidungen über reale physische Objekte und Szenen auf der Grundlage von wahrgenommenen Bildern zu treffen" (Shapiro & Stockman, 2001)

Zu den wichtigen zentralen Aufgaben in CV zählen Objektdetektion und -klassifikation, Objektverfolgung und Szenenrekonstruktion.

In den letzten Jahren hat ML, insbesondere die Deep-Learning-Technik, schnelle Fortschritte in diesem Forschungsgebiet ermöglicht. So wurden neben den klassischen CV- Ansätzen auch verstärkt tiefe neuronale Netze (Deep Neural Networks, DNN) eingesetzt und damit beachtliche Erfolge erzielt, speziell wenn große Mengen an Trainingsdaten zur Verfügung stehen.

Deep Learning, zu Deutsch „tiefgehendes Lernen“, ist eine Technik der Informationsverarbeitung, die künstliche neuronale Netze (Artificial Neural Networks, ANNs) sowie große Datenmengen zur Entscheidungsfindung nutzt. ANNs orientieren sich an der Funktionsweise des menschlichen Gehirns, das Informationen über ein

---

<sup>8</sup> Vgl. Prince (2012), S. 1ff.

Netzwerk von Neuronen verarbeitet. Ein neuronales Netz besteht aus mehreren hintereinander liegenden Schichten (Layers) von künstlichen Neuronen, die unterschiedliche Funktionen erfüllen. Die einzelnen Neuronen sind nicht kompliziert aufgebaut, doch wenn sie passend kombiniert werden, entsteht aus ihnen ein komplexes Netz, das in der Lage ist, auch große Mengen an Daten zu analysieren und schwierige Probleme zu lösen. Die Informationen werden durch die Eingabeneuronen (Eingabeschicht) aufgenommen, durch die Ausgabeneuronen (Ausgabeschicht) ausgegeben. Die Neuronen, die dazwischen liegen bilden die so genannten verdeckten Schichten („hidden layers“), die die erhobenen Daten verarbeiten und Zusammenhänge und Beziehungen daraus extrahieren. Je komplexer ein Problem ist, desto mehr verdeckte Schichten werden meist in ein Netz eingefügt und desto tiefer ist dementsprechend dieses Netz. Aus der Tiefe der Netze folgt auch der Begriff „Deep Learning“. DNNs sind eine spezielle Klasse von Optimierungsmethoden von ANNs. Der wesentliche Unterschied besteht in der Anzahl der verdeckten Schichten. DNNs versetzen Computer in der Lage, aus eigenen Erfahrungen zu lernen. Die Bauweise der DNNs ermöglicht es, auf Basis neuer Informationen, das Erlernte immer wieder mit neuen Inhalten zu verknüpfen und zu erweitern. DNNs verbessern die Vorhersageleistung und haben die Grenzen des Möglichen erweitert. Sie ermöglichen es Probleme zu lösen, die mit klassischen Methoden nicht lösbar sind. Sie werden im Bereich der digitalen Bildverarbeitung verwendet, um schwierige Probleme wie Klassifizierung, Segmentierung und Erkennung zu lösen.

Zu Beginn basierte das Forschungsfeld der CV jedoch auf traditionellen Vorgehensweisen. Im Allgemeinen funktionieren traditionelle CV-Ansätze derart, dass sie die Merkmale eines Bildes (Farben, Kanten, Ecken und Objekten) extrahieren und in Vektoren gruppieren, um anschließend die Merkmalsvektoren mit bekannten Objekten auf wahrscheinliche Übereinstimmungen zu vergleichen. Dabei werden Merkmalsdiskreptoren (SIFT, SURF, BRIEF, etc.) für Bildklassifizierung und Objekterkennung genutzt.<sup>9</sup> Diese Diskreptoren sind dadurch charakterisiert, dass Menschen Merkmale manuell auswählen, die ihrer Meinung nach den relevanten Merkmalen einzelner Objekte darstellen (Feature Engineering), um die

---

<sup>9</sup> Vgl. Mahony et al., (2020), S. 7f.

Vorhersageleistung des Algorithmus zu erhöhen. Zu diesem Zeitpunkt waren Bilder im Vergleich zu heute in einer wesentlich geringeren Anzahl, schlechteren Qualität und Zugreifbarkeit vorhanden<sup>10</sup>. CV hat im Jahr 2012 unter Zuhilfenahme von tiefen neuronalen Netzen, einen der bisher größten Schritte zu einer verbesserten Modellperformance in verschiedenen Dimensionen gemacht und damit auch außerhalb der Wissenschaft mehr Nachfrage generiert. Im Jahr 2012 erzeugten Krizhevsky et al. ein Modell, dessen Training auf DL basierte und deren Genauigkeit der vorhergesagten Labels eine sprunghafte Steigerung aufwies. Dies führte dazu, dass diese Veröffentlichung, zu den aktuell meistzitierten Arbeiten im Bereich von CV zählt.<sup>11</sup> Die weiteren Jahre zeichneten sich durch weitere Anstiege der Vorhersagegenauigkeit aus, allerdings wuchsen die Parameter und Modellgröße mit, sodass auch der Ressourcenverbrauch stark gestiegen ist<sup>12</sup>. Als Gegenbewegung zeichneten sich DNNs ab, welche eine deutlich verringerte Parameteranzahl und Modellgröße aufwiesen<sup>13</sup>. State of the Art ist ein DNN, welches sich selbst trainiert<sup>14</sup>. Alle DNNs messen sich an dem 2009 erstellten ImageNet<sup>15</sup>, welcher als Challenge für neue Veröffentlichungen und deren Vergleichbarkeit untereinander genutzt wird.

DL-Ansätze haben sich als eines der beliebtesten Werkzeuge zur Durchführung von CV Aufgaben etabliert. Im Vergleich zu herkömmlichen Computer-Vision-Techniken ermöglichen diese neuen Ansätze des DL für CV eine höhere Genauigkeit<sup>16</sup> und sie erreichen Ergebnisse, die bisher nur Menschen vorbehalten schienen<sup>17</sup>. Der Hauptunterschied der DL-Ansätze für CV ist das Konzept des End-to-End-Learnings, bei dem einem Netzwerk Rohdaten und eine Aufgabe, wie Klassifizierung, zugewiesen werden, und das Netz lernt, dies automatisch auszuführen. Es ist nicht mehr erforderlich, die Features zu definieren und Feature-Engineering durchzuführen. Diese Schritte werden von den DNNs übernommen.<sup>18</sup> Die relevanten Merkmale werden nicht

---

<sup>10</sup> Vgl. Mahony et al. (2020), S. 7f.

<sup>11</sup> Vgl. Krizhevsky et al. (2012), S. 1

<sup>12</sup> Vgl. Xin et al. (2019), S. 318

<sup>13</sup> Vgl. Canziani et al. (2017), S. 1ff.

<sup>14</sup> Vgl. Xie et al. (2020), S. 1

<sup>15</sup> Vgl. Deng et al. (2009), S. 248ff.

<sup>16</sup> Vgl. Wu et al. (2015), S. 1ff.

<sup>17</sup> Vgl. He et al. (2015), S. 1026ff.

<sup>18</sup> Vgl. Mahony et al. (2020), S. 3f.

vortrainiert, sondern sie werden selbständig gelernt, während das Netz anhand einer Menge von Bildern trainiert wird. Dank aktueller Fortschritte ist das DL inzwischen so weit, dass es einige Aufgaben besser bewältigt als Menschen, etwa die Klassifikation von Objekten auf Bildern. Deep-Learning-Modelle können manchmal genauere Ergebnisse erzielen als Menschen.

Trotz ihrer beeindruckenden Leistung haben Forschungen<sup>19 20 21</sup> auf die Anfälligkeit von DNN-Modellen für sogenannte „Adversarial Examples“ hingewiesen, zu Deutsch „gegensätzliche Beispiele“. Adversarial Examples sind Bilder, die die Computer-Wahrnehmung bewusst täuschen sollen und die ML-Modelle zu einer Falschklassifizierung bringen. Dazu werden die Bilder leicht modifiziert. Einzelne Pixel werden so verändert, dass die Änderungen für das menschliche Auge möglicherweise nicht sichtbar sind. Allerdings führen diese Änderungen dazu, dass die Klassifizierungsvorhersage des ML-Modells mit hoher Sicherheit falsch ist. Diese Art möglicher feindlicher Angriffe (adversarial attacks) auf DNN-Modelle bedrohen die Funktionsfähigkeit von DL-Algorithmen zur Bild- und Objekterkennung. Diese könnten beispielsweise autonomen Fahrzeugen große Probleme bereiten, wenn beispielsweise Verkehrszeichen falsch klassifiziert werden und folglich zu einer unerwünschten Fahrzeugsteuerung führen. Die Anfälligkeit für Adversarial Examples macht es schwierig, DNNs in sicherheitskritischen Bereichen anzuwenden.

## 2.2 Verwandte Arbeiten

Die Existenz von Adversarial Angriffe und die Gefahr, die sie mit sich bringen, haben in den letzten Jahren erhebliche Forschungsaufmerksamkeit erhalten. Es ist davon auszugehen, dass die meisten bisherigen Adversarial Attacks für Forschungszwecke durchgeführt und publiziert wurden. Die zunehmende Verbreitung von KI in verschiedensten Anwendungsfeldern wird das Risiko tatsächlicher Täusch- und

---

<sup>19</sup> Vgl. Illyas et al. (2019), S. 1ff.

<sup>20</sup> Vgl. Elsayed et al. (2020), S. 1ff.

<sup>21</sup> Vgl. Stutz et al. (2019), S. 1ff.

Störversuche allerdings auch in kommerziellen Bereichen erhöhen. Dies erzeugt ein ernstes Bedrohungspotential für die Vertrauenswürdigkeit der KI-Algorithmen.

Viele Forscher beschäftigten sich damit, Adversarial Examples zu konstruieren, sowie Methoden zu entwickeln, die Klassifikatoren robuster gegen solche Störungen machen. Hosseini<sup>22</sup> untersuchten die Robustheit der Google Cloud Vision-API gegen Gauß- und Impulsrauschen. Bei der Untersuchung wurde festgestellt, dass das Hinzufügen eines durchschnittlichen Impulsrauschens von 14,25% ausreicht, um die API zu täuschen. Dodge und Karam<sup>23</sup> untersuchten, wie sich Bildqualitätsverzerrungen auf die Erkennungsleistung auswirken. In der Forschung wurde gezeigt, dass Ansätze, die auf DNNs basieren, anfällig für Verschlechterungen sind, insbesondere für Unschärfe und Rauschen. Zhou et al.<sup>24</sup> untersuchten die Auswirkungen von Unschärfe und Rauschen auf Bildklassifizierer, die auf DNNs basieren und zeigten auch ihre Schwachstellen. Deren Ergebnisse deuten darauf hin, dass die Umschulung (re-training) und Feinabstimmung (fine-tuning) der Modelle mit verrauschten Bildern einen großen Effekt erzielen kann und besser als ein erneutes Training ist. Temel et al.<sup>25</sup> untersuchten die Erkennungsleistung von Verkehrszeichen häufig verwendeter Methoden unter realistischen anspruchsvollen Bedingungen und zeigten, dass Unschärfe, Belichtung und Codec-Fehler direkt dazu führen, dass formbezogene Informationen verloren gehen oder falsch platziert werden. In ähnlicher Weise untersuchten Das et al. die Anfälligkeit der DNNs für Adversarial Examples. Das et al.<sup>26</sup> demonstrierten wie eine systematische JPEG-Komprimierung als effektiver Vorverarbeitungsschritt in der Klassifizierungspipeline wirken kann, um Adversarial Examples entgegenzuwirken und ihre Auswirkungen drastisch zu reduzieren.

Bei viele der oben beschriebenen Untersuchungen wurden die Angriffe unter idealen Bedingungen durchgeführt. In diesem Fall hatten die Forscher vollständige Kenntnis der Architektur und der Modellparameter des tiefen Neuronalen Netzes, was als White-Box-Angriff bezeichnet wird. Nur wenige Autoren untersuchten die Szenarien, bei denen der

---

<sup>22</sup> Vgl. Hosseini et al. (2017), S. 1ff

<sup>23</sup> Vgl. Dodge und Karam (2016), S. 1ff.

<sup>24</sup> Vgl. Zhou et al. (2020), S. 1ff.

<sup>25</sup> Vgl. Temel et al. (2017), S. 1ff.

<sup>26</sup> Vgl. Das et al. (2017), S. 1ff.



Angreifer keine Kenntnis der Modellparameter und der Architektur des Neuronalen Netzes hat (Black-Box-Angriff).<sup>27 28 29</sup> Diese Unkenntnis vermindert die Erfolgswahrscheinlichkeit von Angriffen erheblich, macht sie aber nicht unmöglich wie verschiedene Untersuchungen zeigen.

Große Cloud-Anbieter wie Google, AWS und Microsoft bieten verschiedene Computer Vision-bezogene Dienste an, darunter Bildklassifizierung, Objektidentifizierung, Bilderkennung usw. Diese plattformbasierten Dienste bieten eine verwaltete Plattform für ML, mit der Nutzer möglichst schnell und einfach ihre eigenen Modelle trainieren und einsetzen können. Sie stellen ihren Kunden gegen Gebühr DL-Schnittstellen (APIs) zur Verfügung, mit denen sie CV-Aufgaben ausführen können. Dabei steht eine Auswahl an unterschiedlichen Frameworks und Algorithmen zur Verfügung. Dadurch entfällt der Mehraufwand durch die Verwaltung eigener Hardware. Der Einsatz von Cloud Lösungen für DL ermöglicht die einfache Erfassung und Verwaltung großer Datensätze zum Trainieren von Algorithmen sowie die effiziente Skalierung von DL-Modellen. Dadurch können Nutzer auch ohne Fachkenntnisse komplexe neuronale Netze schnell entwerfen, entwickeln und trainieren wodurch sich die Cloud Lösungen zunehmender Beliebtheit erfreuen.

Während einige Untersuchungen ausführlich gezeigt haben, dass DL-Klassifizierungsmodelle für Adversarial Examples anfällig sind, erhalten Cloud-basierte Bilderkennungsmodelle, die ähnliche Sicherheitsbedenken haben und als Black-Box Systeme betrachtet werden können, bisher noch nicht genügend Aufmerksamkeit.

Unternehmen weltweit setzen auf Cloud Services. Aus diesem Grund wird eine umfassendere Robustheitsanalyse der Bildklassifizierung-Services gebraucht. Um diese Lücke zu schließen, betrachten wir in unserer Arbeit die Bildklassifizierungsdienste der drei wichtigsten Cloud-Plattformen: AWS, Google Cloud und Azure. Wir wollen die Services anhand ihrer Performance und Handhabung vergleichen und untersuchen wie

---

<sup>27</sup> Vgl. Juuti et al. (2019), S. 1ff.

<sup>28</sup> Vgl. Elsayed et al. (2018), S. 1 ff.

<sup>29</sup> Vgl. Li et al. (2019), S. 1 ff.

sich die Erkennungsleistung der Algorithmen unter anspruchsvollen Bedingungen wie Bildrauschen und Unschärfe ändert.

### 3 Datensatz

#### 3.1 Problemstellung

Nachdem das Ziel und die grundlegende Projektstruktur festgelegt wurden, galt es nun, die Vorgehensweise bei der Datenbeschaffung zu klären. Auf den reinen Einsatz vorgefertigter Datensets wurde verzichtet, da der Anlernprozess des Modells so flexibel wie möglich gestalten werden sollte. Es wurde entschieden, mit Hilfe eines Webcrawlers verschiedene Webseiten anzusteuern und die für das Modell relevanten Bilder zu extrahieren.

#### 3.2 Webcrawling

Grundsätzlich werden beim Webcrawling mit Hilfe sogenannter Spiders Webseiten durchsucht und analysiert. Der Anlass bzw. das Ziel dieser Analysen können dabei stark variieren. Einer der ersten Webcrawler war der sogenannte „World Wide Web Wanderer“.<sup>30</sup> Dieser wurde Anfang 1993 von Matthew Grey programmiert und diente dazu, Statistiken über die Entwicklung des Internets zu erstellen. Darüber hinaus wurden alle Seiten in einem zentralen Index zusammengetragen, dem sogenannten „Wandex“, welcher als erste richtige Suchmaschine angesehen wird.

Webcrawler können aber auch für andere Zwecke eingesetzt werden, wie beispielsweise das Sammeln von eMail-Adressen oder, wie für den Sachverhalt benötigt, das Identifizieren und Speichern von Bildern.

---

<sup>30</sup> Vgl. Alpar et al, (2015) S. 37

### 3.3 Wahl des Werkzeugs

Im Laufe der Zeit wurden immer mehr und mehr Frameworks und Programme zum Crawlen von Webseiten entwickelt. Die Art und Weise, wie diese verwendet und implementiert werden, unterscheidet sich genauso, wie die jeweiligen Vor- und Nachteile. Die Software „Helium Scraper“ bietet beispielsweise ein sogenanntes Point & Click Interface, wodurch die Erstellung einer Spider auch für Nutzer ohne Programmier- oder Webentwicklungskenntnisse einfach durchzuführen ist.<sup>31</sup> Die Kosten für eine Einzelnutzerlizenz mit Updatesupport für drei Monate belaufen sich hierbei auf 99 Dollar, während eine Enterpriselizenz mit Unterstützung für zehn Benutzer schon mit 699 Dollar zu Buche schlägt.<sup>32</sup> Die Software „OutWit Hub“ hingegen ist beispielsweise kostenlos, scheitert allerdings an der Lösung von Captchas.<sup>33</sup> Software as a Service Anbieter wie „Visual Web Ripper“ bewältigen Captchas und andere Herausforderungen wie dynamisch nachladende Seiten, verwenden allerdings ein Abomodel mit jährlichen Kosten von 349 Dollar für einen einzelnen Benutzer.<sup>34</sup>

Nachdem verschiedene Anbieter und Werkzeuge miteinander verglichen wurden, fiel die Entscheidung auf das Scrapy Framework.<sup>35</sup> Dieses ist ein open-source Projekt, welches seit 2008 entwickelt wird. Es wurde in Python entwickelt und ist plattformunabhängig einsetzbar.<sup>36</sup> Durch die vielseitigen Features und die einfache Art, es zu verwenden, hat sich im Laufe der Jahre eine große Community gebildet. Darüber hinaus ist das Framework sehr gut dokumentiert und kostenfrei zu verwenden.

### 3.4 Erstellung der Crawler

Nachdem Scrapy als Werkzeug gewählt und die Datenquellen bestimmt wurden, galt es nun, die verschiedenen Spiders zu entwickeln. Bei Scrapy wird eine Spider durch die Implementierung der generischen Klasse „scrapy.spiders.spider“ realisiert.<sup>37</sup> Eine Spider

---

<sup>31</sup> Vgl. Helium Software, (2020)

<sup>32</sup> Vgl. Helium Software, (2020)

<sup>33</sup> Vgl. OutWit Technologies, (2020)

<sup>34</sup> Vgl. Sequentum, (2020)

<sup>35</sup> Vgl. Kouzis-Loukas (2016), S.1ff

<sup>36</sup> Vgl. ebd, S. 2

<sup>37</sup> Vgl. ebd, S. 42

verfügt immer über die beiden Attribute „name“ und „allowed\_domains“. Ersteres dient dazu, die Spider zu identifizieren und anzusteuern. Das zweite Attribut schränkt ein, welche URLs Teil des Crawlprozesses sein sollen. Dadurch lässt sich vermeiden, dass beliebig tief bzw. breit gecrawled wird.

Darüber hinaus muss jede Spider zwei Methoden implementieren. Die Methode „start\_requests“ stellt den ursprünglichen Startpunkt des Crawlens dar. Über einen Aufruf von „scrapy.Request“ wird der Prozess gestartet. Bei diesem Aufruf werden zwei Parameter übergeben. Zum einen die zu crawlende Adresse und zum anderen den Methodennamen „parse“. Dabei handelt es sich um die zweite zu implementierende Methode. Diese erhält die Antwort der Website, welche auch „reponse“ genannt wird. Innerhalb dieser reponse findet man Daten zum Header und Body der Website. Diese lassen sich nun mit verschiedenen Werkzeugen und Hilfsmitteln des Scrapy Frameworks analysieren und verarbeiten.

Um die Anforderungen für den Use Case zu erfüllen, musste die Klasse um weitere Attribute erweitert werden. Um zu vermeiden, dass das gleiche Bild mehrmals erfasst wird, wurde ein Set erstellt, indem der Name aller bereits gecrawlten Bilder enthalten ist. Bei jedem Bild wird zunächst geprüft, ob es in dem Set bereits enthalten ist. Ist dies der Fall, wird die Bearbeitung des Bildes übersprungen. Fehlt das Bild bisher im Set, wird es diesem hinzugefügt und anschließend bearbeitet.

Um festzulegen, für welche Tiere die Datenbeschaffung durchgeführt werden soll, wurde eine Textdatei namens „animals“ erstellt, welche alle Tiere beinhaltet, mit denen das Model trainiert werden soll. Darüber hinaus wurde die Klasse um das Attribut „currentAnimal“ ergänzt. Dieses speichert den Namen der Tiergattung, welches gerade gecrawled wird. Der Crawlprozess wird einmal für jede Datenquelle für jedes Tier durchgeführt. Das Attribut „currentAnimal“ wird dafür verwendet, den aktuellen Namen beim Abspeichern der Bilder zu verwenden. Ergänzt wird der Dateiname um den aktuellen Zählerstand, welcher in dem Hilfsattribut „Counter“ gespeichert und nach jedem gespeicherten Bild um eins erhöht wird. Der Name der gecrawlten Seite wird ebenfalls im Dateinamen hinterlegt, um später nachvollziehen zu können, welches Bild von welcher Seite stammt.

Scrapy erlaubt es, mit Hilfe von XPath die aufgerufene Webseite weiter zu bearbeiten. Dadurch war es möglich, gezielt alle als Bild getaggten Elemente anzusprechen und in einer Schleife abzuarbeiten. Im Vorfeld wurde jede Datenquelle hinsichtlich der Art und Weise analysiert, wie die Bilder hinterlegt werden bzw. woran man die Tierfotos von ungewollten Bildern, wie beispielsweise Menüelementen, unterscheiden kann. So enthalten zum Beispiel bei Pikwizard alle Tierbilder den Begriff „photos“ in der Adresse. So lassen sich nach und nach ungewünschte Bilder ausschließen, damit zum Schluss nur noch die relevanten Tierfotos übrigbleiben.

Da diese nicht nur erkannt, sondern auch gesichert werden sollen, wurde die Klasse um eine weitere Methode erweitert, welche als Parameter eine URL erhält und das dahinter vorgefundene Bild abspeichert.

Beim Crawlen der verschiedenen Datenquellen entstanden verschiedene Herausforderungen. Um zu vermeiden, dass der Benutzer sich durch mehrere Seiten klicken muss, verwenden moderne Webseiten dynamisches Nachladen, um durchgängig neue Bilder nachzuladen. So kann der Nutzer beliebig weit scrollen. Umgesetzt wird dies durch einen Ajax Request, welcher ausgelöst wird, sobald sich die Ansicht dem Ende der Seite nähert. Dies wurde beim Crawlen mit Scrapy nicht ausgelöst, da lediglich die ursprüngliche Ansicht der Webseite empfangen und verarbeitet wird. Dies hat zur Folge, dass die Spider je nach Datenquelle nur ein kleines Set an Bildern erhält.

Ein weiteres Problem äußerte sich dadurch, dass beim Versuch eine Seite zu crawlen, der http-Statuscode 403 zurückgegeben wurde. Dieser bedeutet, dass die Anfrage aufgrund fehlender Berechtigung nicht durchgeführt wurde. Der Grund dafür liegt in der Art und Weise, wie Webcrawler sich identifizieren. Bei einem Aufruf einer Webseite werden Informationen über den sogenannten User Agent mitgesendet. Bei einem User Agent handelt es sich um die Schnittstelle bzw. das Programm, mit dem ein Anwender auf Netzwerkdienste zugreift. Ein typisches Beispiel hierfür wäre ein Webbrowser. Teil dieser Informationen ist der Header, welcher detailliert Auskunft darüber gibt, welches Programm und welches Betriebssystem verwendet wird. Wird nun eine der Datenquellen über den Browser aufgerufen, erkennt dies die Webseite als einen regulären und zulässigen Zugriff und beantwortet die Anfrage. Ein Webcrawler gibt sich standardmäßig über den Header als Bot zu erkennen. Da falsch konfigurierte

Webcrawler Performanceprobleme auslösen können, werden Anfragen von jenen blockiert. Um dieses Problem zu umgehen, lassen sich die Headerinformationen beliebig ändern. So ist es möglich, dass der Crawler sich als ein Webbrowser identifiziert und somit nicht ausgesperrt wird. Gemäß des „Crawl Responsibly“-Ansatzes ist es hierbei allerdings umso wichtiger, dass die Anzahl der parallelen Aufrufe und die Zeit zwischen den Aufrufen so gewählt werden, dass der Webserver nicht überlastet wird.<sup>38</sup>

### 3.5 Bildmodifikation

Nachdem die verschiedenen Crawler die ausgewählten Tierbilder erfolgreich erkannt, heruntergeladen und abgelegt haben, mussten anschließend noch verschiedene Bildmodifikationen durchgeführt werden, bevor die Bilder als Trainingsdatensatz verwendet werden konnten.

Die ersten Versuche, das Model mit mehr Bildern zu trainieren schlugen fehl. Kommentiert wurde dies mit der Fehlermeldung hier exemplarisch bei AWS Image Recognition Custom Labels „The manifest file contains too many invalid data objects“. Detailliertere Informationen waren leider nicht zu entnehmen. Dieses Problem betraf alle verwendeten Cloud-Services. Um die Problematik dennoch weiter einzugrenzen, wurde der Trainingsdatensatz immer weiter verkleinert. Als Ergebnis dieser Tests stellte sich heraus, dass das Problem durch Trainingsbilder mit sehr hoher Auflösung verursacht wurde. Jeder Cloud-Service hat hierbei seine eigenen Limitierungen hinsichtlich der Bildauflösung. Um dieses Problem zu lösen, wurde ein Pythonskript als weitere Vorverarbeitungsphase implementiert, welches nach dem erfolgreichen Download der Dateien diese hinsichtlich ihrer Auflösung prüft und gegebenenfalls herunterskaliert. Hierfür musste zunächst eine geeignete Pythonbibliothek gefunden werden, welche die Möglichkeit bietet, Bilder zu transformieren, ohne dass die Bildqualität spürbar darunter leidet. Darüber hinaus war wichtig, dass sich kein Aliasing, also der Effekt der Treppenbildung an Kanten, bildet. Ebenso war essentiell, dass das Seitenverhältnis

---

<sup>38</sup> Vgl. Sitebulb Limited, (2020)

beibehalten wird, da es ansonsten zu Verzerrungen kommt und die Objekte auf den Bildern nicht mehr korrekt dargestellt werden.

Die Python-Bibliothek „SKIMAGE“ stellt mit der Methode „resize“ alle grundlegenden Funktionen zur Verfügung.<sup>39</sup> Mit der Methode „shape“ lässt sich abfragen, welche Auflösung das Bild hat. Mit den Methoden „imread“ und „imsave“ lassen sich die zu bearbeitenden Bilder einlesen und abspeichern.

Das Skript prüft für jedes gecrawlte Bild, wie groß die Auflösung ist. Wenn eine der beiden Bildseiten einen festgelegten Wert übersteigt, wird das Bild angepasst. Als Grenze für die Bildbreite bzw. -höhe wurden 2000 Pixel gewählt, da bei den Tests alle Cloud-Services mit dieser Größe umgehen konnten. Wird also dieser Wert überschritten, prüft der Algorithmus, welche Seite größer ist. Die größere Seite wird anschließend durch 2000 geteilt. Das Ergebnis ist der Faktor, um den sowohl die Bildbreite als auch -höhe geteilt werden, um auf die gewünschte Zielauflösung zu kommen. Beim Aufruf der „resize“ Funktion wird als Parameter „anti\_aliasing=True“ gesetzt, um Aliasing beim Resizing zu verhindern.

Diese Bildmodifikationen waren zwingend notwendig, da das Training der Modelle sonst nicht erfolgreich durchgeführt wurde. Als positiver Nebeneffekt lässt sich zudem hinzufügen, dass der Datensatz nach dieser Modifikation eine um rund 80% verkleinerte Größe aufweist. Damit konnten weitergehende Modifikationen beschleunigt und Bandbreite beim Upload gespart werden.

Nach diesen Modifikationen war der Datenkorpus korrekt aufbereitet und konnte anschließend bei den verschiedenen Cloud-Services hochgeladen werden.

## 4 Methodik

Dieses Kapitel beschreibt die Vorgehensweise der Untersuchungen und die Methode zur Erzeugung der Ergebnisse. Der erste Teil beschreibt, welches Cloud Setup für die Untersuchung benutzt wurde, um die Dienste unter möglichst gleichen Bedingungen zu

---

<sup>39</sup> Vgl. van der Walt et al., (2020)

testen. Darauf aufbauend, werden im zweiten Teil einige Methoden zur Erstellung von Adversarial Examples vorgestellt.

Unsere Untersuchung konzentriert sich auf die (semi-) automatisierten ML Services (AutoML) von Amazon, Google und Microsoft. Diese Dienste sollen den Aufbau von eigenen Modellen stark vereinfachen bzw. vollständig automatisieren. Man übergibt die eigenen Trainingsdaten und die Dienste wählen dann die passenden Algorithmen aus, um die Modelle zu trainieren. Man wählt das gewünschte Verfahren aus und der Rest wird von der Cloud-Plattform übernommen. Mit AutoML können hochwertige Modelle trainieren werden, die auf die eigenen Anforderungen zugeschnitten sind. Mithilfe von überwachtem Lernen kann ein Modell mit eigenen Datensätze darauf trainiert werden, relevante Muster und Inhalte in Bildern zu erkennen und die Bilder nach eigenen definierten Bezeichnungen (Labels) zu klassifizieren.

Die Untersuchung wurde in drei Phasen aufgeteilt: Die erste Phase wurde durchgeführt, um ein grundsätzliches Verständnis für die Arbeitsweise und Handhabung der Cloud-Dienste und einen ersten Eindruck ihrer Performance zu erhalten. Als Trainingsdatensatz für die ML-Modelle wurden 1000 qualitativ hochwertige Tierbilder verwendet auf denen jeweils eines von fünf verschiedenen Tieren (Elefant, Zebra, Löwe, Nashorn, Tiger) abgebildet sind. Anschließend wurden die ML-Modelle wiederum mit hochwertigen Tierbildern getestet und die Erkennungsleistung analysiert. In der zweiten Phase wurde die Robustheit der Klassifizierer gegenüber Adversarial Examples untersucht. Die ML-Modelle wurden mit dem Trainingsdatensatz der ersten Phase trainiert. Beim Test der Modelle wurden allerdings Bilder verwendet, die zusätzlich durch künstliches Rauschen und Bildunschärfe gerändert wurden. Die veränderten Testbilder sollen Adversarial Examples darstellen. In der dritten Phase wurde untersucht wie sich ein Training mit zusätzlichen Adversarial Examples auf die Erkennungsleistung der ML-Modellen auswirkt. Dazu wurde der Trainingsdatensatz mit künstlich modifizierten Bildern erweitert (33% mit Unschärfe und 33% mit Rauschen).



	<b>Trainingsdatensatz:</b>	<b>Testdatensatz:</b>
<b>Phase 1</b>	Bilder guter Qualität	Bilder guter Qualität
<b>Phase 2</b>	Bilder guter Qualität	Adversarial Examples (Bilder mit Unschärfe und Rauschen)
<b>Phase 3</b>	Bilder guter Qualität Und Adversarial Examples (Bilder mit Unschärfe und Rauschen)	Adversarial Examples (Bilder mit Unschärfe und Rauschen)

Abbildung 2: Übersicht der Eigenschaften der jeweiligen Phasen

#### 4.1 Experimentelles Setup

In diesem Abschnitt werden die verfügbare Einstellungsmöglichkeiten der AutoML Services skizziert und die von uns gewählte Konfiguration erläutert.

##### Amazon Web Services (AWS):

Amazon Rekognition Image für Bildanalyse und Amazon Rekognition Video, um Videos zu analysieren. Das System erkennt Objekte, Personen, Texte und Situationen anhand bestimmter Merkmale. Amazon Rekognition ist ein vollständig verwalteter Dienst, der alle Schritte der Entwicklung eines Modells unterstützt. Amazon Rekognition bietet mehrere integrierte ML-Algorithmen, die für zahlreiche Anwendungsfälle verwendet werden können. Zur Klassifikation von Bildern wird der Bildklassifikationsalgorithmus verwendet, der eine Multi-Label-Klassifizierung unterstützt. Der überwachter Lernalgorithmus verwendet neuronale Netze, die von Grund auf neu oder mithilfe von Transferlernen trainiert werden können, wenn keine große Anzahl von Trainingsbildern verfügbar ist. Mit Amazon Rekognition Custom Labels ist es möglich benutzerspezifische Objekte und Szenen in Bildern zu identifizieren. Amazon Rekognition Image berechnet jedes Mal eine Gebühr, wenn ein Bild mit der API analysiert wird.

Für die Analyse wurden die Trainings- und auch Testbilder in einem eigenen Amazon S3-Bucket abgelegt. Durch Nutzung einer Ordnerstruktur im S3-Bucket konnten die sich darin befindlichen Bilder mit Labels versehen werden - diese entsprechen dann dem jeweiligen Ordnernamen. Weiterhin kann entweder ein automatischer Train-Test-Split

von 80/20 von genutzt werden oder wie in dem durchgeführten Projekt die Aufteilung des Bilddatensatzes über die beiden S3-Buckets gesteuert werden.

### **Google Cloud Platform (GCP):**

AutoML Vision ist der automatisierte Bilderkennungsservice von Google. Damit können Bilder hochgeladen, gekennzeichnet und dann genutzt werden, um ML-Modelle selbst zu trainieren. Mit AutoML Vision können Informationen aus Bildern in der Cloud oder Edge analysiert werden. In Rahmen dieser Arbeit /Für die Untersuchung wird ein in der Cloud gehostetest Modell verwendet. Um ein in der Cloud gehostetes Modell trainieren zu können, muss zuerst ein Cloud Storage-Bucket erstellt werden, in dem die Trainingsbilder direkt oder auch zu einem späteren Zeitpunkt importiert werden müssen. Um die AutoML Vision API verwenden zu können, müssen sich die Projektressourcen derzeit in der Region „us-central1“ befinden. Als nächstes sollte entschieden werden, ob eine manuelle oder automatische Aufteilung der Daten bevorzugt ist. Die Daten in einem Dataset werden beim Trainieren eines Modells standardmäßig in drei Datasets aufgeteilt: ein Trainings-Dataset, ein Validierungs-Dataset und ein Test-Dataset. Die Bilder, die sich im Trainingsdatensatz befinden werden zum Trainieren des Modells verwendet. Damit werden die Parameter des Modells erlernt, vor allem die Gewichtung der Verbindungen zwischen den Knoten des neuronalen Netzes. Um Verzerrungen im Modell zu vermeiden, werden Validierungs- und Testdatensatz verwendet. Das Validierungs-Dataset wird zum Validieren der Ergebnisse verwendet, die das Modell während des Trainings zurückgibt. Das Test-Dataset wird nach dem Trainieren des Modells zum Prüfen der Ergebnisse des Modells verwendet. Standardmäßig teilt AutoML Vision das Dataset nach dem Zufallsprinzip folgendermaßen auf: 80 % für das Training, 10 % für Validierung und 10 % für die Bewertung des Modells. Die Datenaufteilung könnte auch manuell gesteuert werden, in dem CSV-Datei mit Bild-URLs und -labels erstellt wird. Allerdings ist diese Alternative zeitaufwendig. Der erste Schritt zum Erstellen eines benutzerdefinierten Modells besteht darin, ein leeres Dataset zu erstellen, das mit den Trainingsdaten für das Modell gefüllt wird. Hier muss geben der Klassifizierungstyp angegeben werden, der das benutzerdefinierte Modell ausführen soll.

Die Single-Label Klassifikation weist jedem klassifizierten Bild ein einzelnes Label zu. Bei der Multi-Label Klassifikation können einem Bild mehrere Labels zugewiesen werden. In dieser Arbeit wurde die Multi-Label Klassifikation verwendet.

### **Microsoft Azure:**

Custom Vision ist der Bilderkennungsservice, welcher bei Azure automatisiert die zugeführten Bilder erkennen soll. Zunächst muss eine Ressource erstellt werden, zu der das anwendungsspezifische Projekt zugewiesen werden kann. Bei der Erstellung konnte zudem der Typ "Classification" gewählt werden sowie die Art der Klassifikation – "Multi-Label". Weiterhin wurde die Domäne "General" selektiert, um zum einen den Vergleich mit den anderen Services fair zu gestalten und zum anderen war keine passende Domäne für das aus diesem Projekt vorhandene Klassifikationsproblem vorhanden. Nachfolgend konnten die Bilder beim Hochladen mit dem entsprechenden Label versehen werden, eine Möglichkeit mittels Ordnerstruktur oder CSV-File die Labels zu vergeben war nicht gegeben. Als problematisch gestaltete sich der nicht zu verändernde Train-Test-Split von 80% zu 20%, welcher eine Abweichung zu den beiden anderen Services darstellt. Bis auf die Definition von dem verwendeten Trainingsbudget waren keine weiteren nennenswerten Einstellungsparameter beim Training zur Verfügung stehend.

	<b>Amazon</b>	<b>Google</b>	<b>Microsoft</b>
Automatisierte Modelle	Amazon Rekognition Custom Labels	Cloud AutoML	Custom Vision
Gewählte Modellierungs- methoden	Keine Definition möglich	Multi-Label Klassifikation	Multi-Label Klassifikation
Datenaufteilungs- strategie:	Manuelle Aufteilung	Automatische Aufteilung	Automatische Aufteilung

	<b>Train: 90%</b> <b>Test: 10%</b>	<b>Train: 80 %</b> <b>Validierung:</b> 10% <b>Test:10%</b>	<b>Train: 80%</b> <b>Test: 20 %</b>
Automatisierter Vorschlag für Algorithmus	Ja	Ja	Ja
Bildformat	JPEG PNG	JPEG PNG	JPEG PNG
Daten	999	988 Bilder	997
Modelltyp	Cloud	Cloud	Cloud
Trainingskosten	Keine Definition möglich	16 Knotenstunden	8 Stunden Training Budget

Abbildung 3: Vergleich der verschiedenen Cloud Anbieter

Der größte Nachteil des Angebots ist die Lernkurve und die erforderliche Projektzeit. Trotz der Zugänglichkeit jeder Phase müssen viel Zeit und Mühe investieren werden, um ein Projekt abzuschließen. Zusätzlich wirkt es sich ungünstig aus, dass die Dokumentation in Bezug auf Fehlerfälle knapp gefasst ist. So traten beispielsweise Fehlermeldungen ohne Fehlercode auf, wodurch es kaum nachvollziehbar war, was diese Fehlermeldungen verursacht hat. Diese waren allerdings erst ersichtlich, nachdem das Modelltraining gestartet wurde. Somit ist im Umkehrschluss keine Prüfung seitens Cloud-Services beim Upload der Bilder erfolgt.

#### 4.2 Erkennung unter herausfordernden Bedingungen

In diesem Kapitel werden die verwendeten Methoden zur Erstellung von Adversarial Examples vorgestellt.

#### 4.2.1 Erstellung von Adversarial Examples

Neuronale Netze sind empfindlich gegenüber Unschärfe und Bildrauschen<sup>40</sup>. Unschärfe kann dabei aus verschiedenen Gründen auftreten, wie beispielsweise durch einen falsch eingestellten Fokus. Gerade beim Fotografieren von schnellen Objekten bzw. Tieren kann durch eine zu niedrige Verschlusszeit das Motiv nicht mehr scharf dargestellt werden. Äußere Einflüsse wie Erschütterungen können ebenfalls zu unscharfen Fotos führen. Bildrauschen hingegen entsteht im Regelfall durch eine zu hohe Lichtempfindlichkeitseinstellung des Kamerasensors. Schlechte Lichtverhältnisse bestärken diesen Effekt.

Um das Model also resistenter gegen solche Störungen zu machen, wurde ein weiteres Pythonskript geschrieben, welches die gecrawlten Originalbilder künstlich mit Unschärfe und Bildrauschen versieht. Hierfür wurde die sogenannte „Python Imaging Library“ verwendet, welche verschiedene Imagefilter bietet, mit denen sich Bilder manipulieren lassen. Auf diese Weise ließen sich die Bilder aus dem Datensatz leicht mit Unschärfe und Rauschen versehen.

## 5 Ergebnisse

Im Fokus dieses Kapitels stehen die Ergebnisse der in Kapitel 4 beschriebenen Untersuchungsphasen. Zur Einsicht der Ergebnisse wurden Säulendiagramme erstellt, welche die Treffgenauigkeit (Accuracy) der Modelle darstellen (Y-Achse). Diese Metrik gibt den Anteil an richtigen Vorhersagen an. Je nach Thema gilt ein Modell in der Regel bereits als sehr gut, wenn die Genauigkeit zwischen 85 und 90 Prozent liegt. Auf der X-Achse werden die Cloud-Services angezeigt. Hierbei wurde bei allen Grafiken die Y-Achse auf den Wertebereich über 0,9 (90%) beschränkt, um die Unterschiede zwischen den Modellen besser sichtbar machen zu können.

---

<sup>40</sup> Vgl. Dodge und Karam, (2016), S. 1ff.

### 5.1 Ergebnisse Phase 1

In der ersten Phase wurden die AutoML Services anhand ihrer Handhabung und Performance verglichen. Bei den Trainings- und Testdaten wurden nur Originalbilder ohne zusätzliches Rauschen oder zusätzliche Unschärfe verwendet. Insgesamt zeigt sich, dass alle drei Services bei qualitativ guten Bildern eine sehr gute Erkennungsleistung liefern (Accuracy = 100%). Dabei sollte darauf hingewiesen werden, dass der Datensatz relativ klein ist (5 Label à 180 Bilder) und nur eine beschränkte zugewiesene Rechenkapazität verwendet wurde.

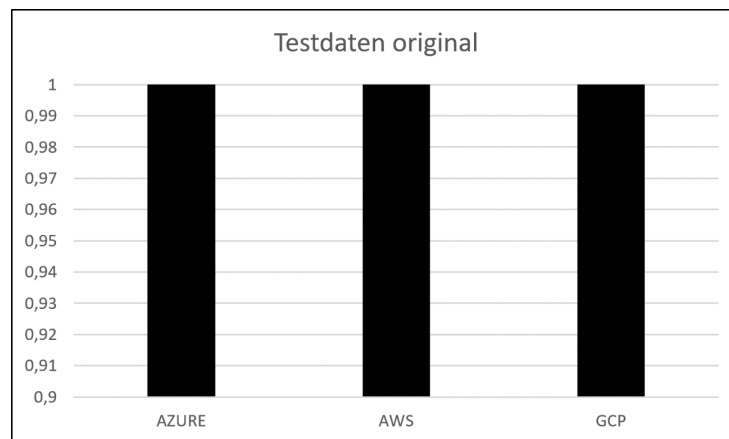


Abbildung 4: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 1

### 5.2 Ergebnisse Phase 2

In der zweiten Phase wurde die Robustheit der Modelle gegenüber Adversarial Examples getestet, in die Bilder verwendet wurden, die zusätzlich durch künstliches Rauschen und Bildunschärfe modifiziert wurden.

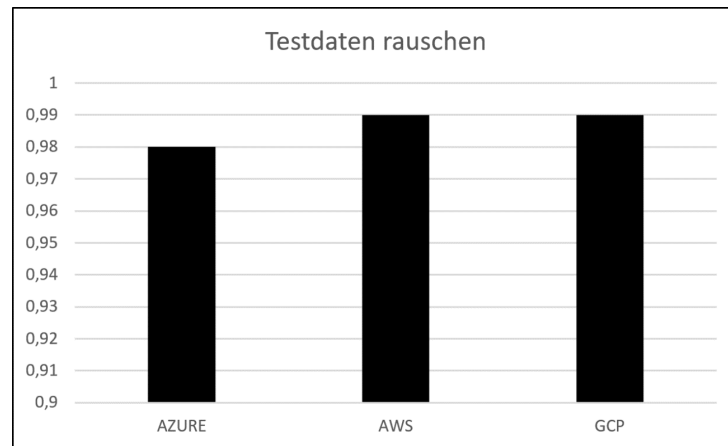


Abbildung 5: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 2 (Rauschen)

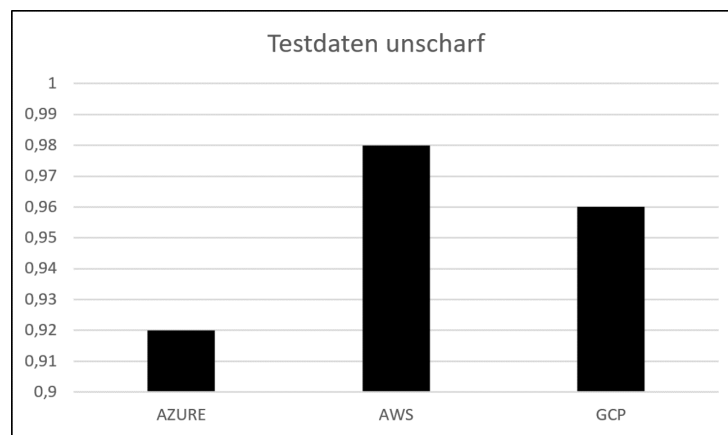


Abbildung 6: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 2 (Unschärfe)

Bei der Beobachtung des Säulendiagramms (Siehe Abbildung 5) fällt auf, dass das künstlich hinzugefügte Rauschen nur zu einer geringen Verschlechterung der Genauigkeit geführt hat. Das Modell von Azure hat mit 98% Accuracy am schlechtesten abgeschnitten. Darauf folgen AWS und Google, welche mit 99 % Accuracy dicht beieinander liegen.

Im Vergleich zum Rauschen führt die künstlich hinzugefügte Unschärfe zu einer größeren Verschlechterung der Treffgenauigkeit. (Siehe Abbildung 6) Dabei erzielte AWS mit 98% Accuracy den höchsten Wert. Google und Azure zeigen Schwächen bei der Erkennung von unscharfen Bildern.

Zwar weisen die Modelle immer noch eine gute Treffgenauigkeit auf, ein Blick auf Abbildung 7 gibt allerdings Aufschluss darüber, dass die Erkennungsleistung einiger Modelle durch Verrauschen und unscharfe Bilder stark abnimmt. Die Vorhersagesicherheit des Googles Modells sinkt im gezeigten Beispiel von 91% auf 59%. Dies deutet drauf hin, dass das Modell nicht robust gegen Adversarial Examples ist.




			
	<i>Original</i>	<i>Rauschen</i>	<i>Unschärfe</i>
<b>Amazon Rekognition</b>	Nashorn ( 99%)	Nashorn ( 74%)	Nashorn ( 63%)
<b>Google AutoML Vision</b>	Nashorn ( 91%)	Nashorn ( 59%)	No Objects
<b>Azure Custom Vision</b>	Nashorn ( 100%)	Nashorn ( 100%)	Lion ( 100%)

Abbildung 7: Erkennungsgenauigkeit der Cloud Anbieter nach Bildmodifikationen

### 5.3 Ergebnisse Phase 3

In der dritten Phase wurde untersucht wie sich ein Training mit zusätzlichen Adversarial Examples auf die Erkennungsleistung der ML-Modellen auswirkt.



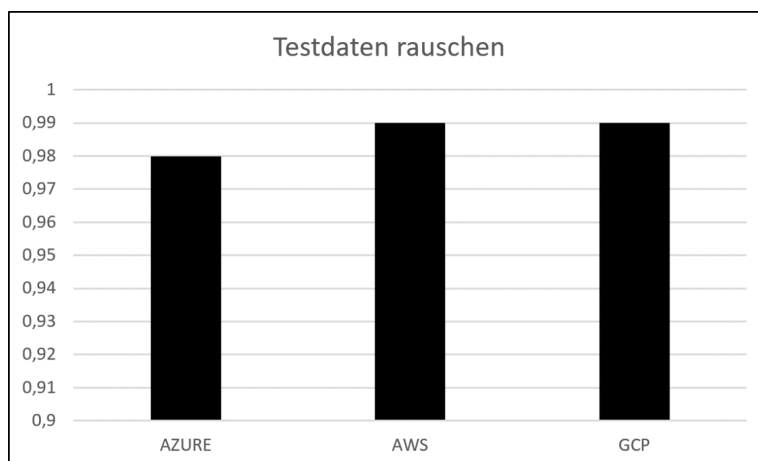


Abbildung 8: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 3 (Rauschen)

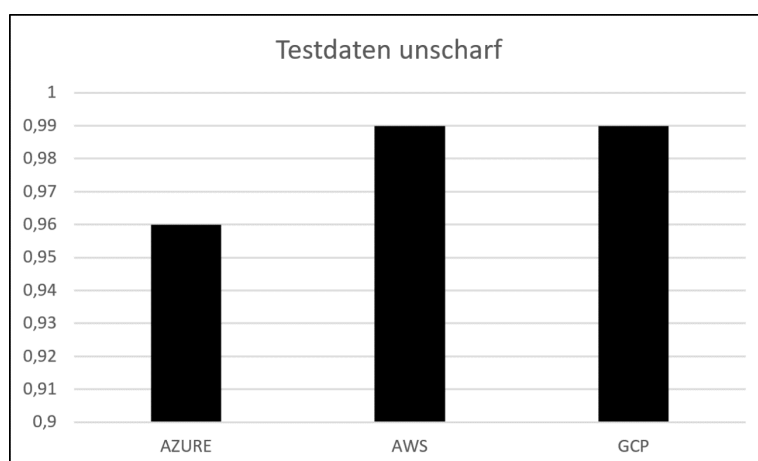


Abbildung 9: Erkennungsgenauigkeit der Cloud Anbieter nach Phase 3 (Unschärfe)

Ein Vergleich der Abbildungen 8 und 9 zeigt, dass durch die zusätzliche Verwendung von verrauschten Adversarial Examples beim Training keine Verbesserung erreicht wurde. Bei der Treffgenauigkeit der Modelle ist keine Änderung zu erkennen. Im Gegensatz dazu, führt die Verwendung von unscharfen Bildern im Training zu einer Verbesserung der Accuracy. Allerdings sollte beachtet werden, dass die Modelle auch ohne Training mit verrauschten Bildern eine sehr hohe Accuracy aufweisen.

Zusammenfassend ist AWS als CV Cloud-Service für den geschilderten Anwendungsfall am besten geeignet. Zum einen wird die für den Anwendungsfall höchste Accuracy ausgewiesen. Zum anderen erwies sich der Umgang mit Test- und Trainingsdaten einfacher als bei den Konkurrenzdiensten. Die genutzte Methode die Test- und

Trainingsdaten in S3-Buckets zu definieren und mit der dort intuitiv möglichen Ordnerstruktur zum Labeling stellte sich als zeitsparend heraus. Zudem waren Ergebnisse reproduzierbar und Testdaten wurden nicht randomisiert ermittelt. Weiterhin waren zum Versuchszeitpunkt mehr Möglichkeiten zur Konfiguration gegeben, die sich theoretisch bei anderen Anwendungsfällen als nützlich erweisen könnten.

## 6 Diskussion

Im Zuge dieser Ausarbeitung wurde ein erstes, funktionierendes Model entwickelt, welches zuverlässig verschiedene Tiere erkennen und benennen kann. Um den Nutzen des Modells weiter zu steigern, lassen sich verschiedene weitere Features benennen, die jedoch den Rahmen dieses Projekts überschritten hätten.

Alle Tests basierten auf der grundlegenden Annahme, dass lediglich ein Tier auf dem Bild zu erkennen ist. In der Realität wird dies bei Wildtierkameras in Abhängigkeit des zu beobachtenden Tieres, insbesondere bei Herdentieren, selten der Fall sein. Aus diesem Grund wäre es notwendig, per Object Detection zunächst die verschiedenen Tiere zu erkennen und zu separieren, um dieses dann anschließend zu klassifizieren. Mit Hilfe von Image Segmentation wäre es möglich, das Bild zunächst zu segmentieren und so besser analysieren zu können. Je nach Anwendungsfall wäre es möglich, mit Behavioral Classification zu erkennen, was die Tiere tun bzw. wie sie sich verhalten. Das könnte zum Beispiel Aufschluss über den Gesundheitszustand geben.

Um noch mehr Informationen zu erhalten, könnten die Kamerafallen mit weiteren Sensoren ausgerüstet werden, welche Daten über Temperatur, Luftfeuchtigkeit, Windgeschwindigkeiten oder Wasserpegel sammeln. So lassen sich gegebenenfalls Muster oder Auffälligkeiten erkennen, welche wiederum bestimmte Verhaltensweisen erklären könnten.

Die Trainingsdaten bestanden primär aus hochqualitativen Fotoaufnahmen. Da es sich bei Wildtierkameras selten um hochauflösende Spiegelreflexkameras mit perfekten

Einstellungen handelt, wäre ein denkbarer nächster Schritt, die Modelle mit qualitativ schlechteren Aufnahmen zu trainieren, um diese robuster zu machen.

## 7 Fazit

Zu Beginn des Projekts wurden drei wesentliche Ziele definiert.

Beim ersten Ziel wurde die Frage gestellt, ob es möglich ist, mit den begrenzten Ressourcen ein Modell zu trainieren, welches sinnvoll in dem Praxiskontext Anwendung finden kann. In Anbetracht der sehr positiven Ergebnisse gemessen an den Erkennungsraten, lässt sich das Ziel im gewählten Versuchsaufbau als erreicht sehen. Alle Cloud-Services weisen eine Genauigkeit von 100% aus und waren somit dazu in der Lage, zuverlässig die Tiere richtig zu erkennen und zu benennen.

Bei dem zweiten Projektziel ging es darum zu untersuchen, welcher der Cloud-Services sich für die Umsetzung am besten eignet. Grundsätzlich gilt für alle Anbieter gleichermaßen, dass der Einstieg in die Thematik und das Einarbeiten relativ aufwändig ist. Darüber hinaus kam erschwerend dazu, dass die Dokumentation hinsichtlich Fehlermeldungen nicht durchgängig gepflegt ist und somit generische Fehlermeldungen dafür sorgten, dass das Anlernen der Modelle abbrach, ohne dass dem Nutzer aufgezeigt wurde, wodurch der Fehler verursacht wurde. Dadurch musste per Trial & Error erkannt werden, dass einige wenige Bilder eine zu hohe Auflösung besaßen, was letztendlich der Grund für den Abbruch war. Bei den Tests ohne Bildmodifikationen erreichten alle Cloud-Services eine Erkennungsrate von beeindruckenden 100%. Erst nachdem die Bilder mit Rauschen und Unschärfe versehen wurden, ergaben sich verschiedene Genauigkeiten. Zusammenfassend lässt sich die Frage des zweiten Projektziels damit beantworten, dass AWS sich insgesamt am besten für die Umsetzung des Sachverhalts anbietet.

Bei dem dritten Projektziel ging es darum, herauszufinden, wie Adversarial Examples die Performance beeinflussen. Im Falle dieser Ausarbeitung ergab sich, dass sowohl AWS als auch GCP mit den Bildmodifikationen keine signifikanten Probleme hatten. Lediglich Azure zeigte eine verringerte Erkennungsrate auf, insbesondere bei unscharfen Bildern.

Zusammenfassend konnten alle Projektfragestellungen analysiert und beantwortet werden.

## Literaturverzeichnis

### Monographien

- AKodagali, Jyoti, und S Balaji. „Computer Vision and Image Analysis Based Techniques for Automatic Characterization of Fruits A Review“. *International Journal of Computer Applications* 50, Nr. 6 (28. Juli 2012): 6–12. <https://doi.org/10.5120/7773-0856>.
- Alpar, Andre, Markus Koczy, und Maik Metzen. *SEO - Strategie, Taktik und Technik: Online-Marketing mittels effektiver Suchmaschinenoptimierung*. Springer-Verlag, 2015.
- OpenAI. „Attacking Machine Learning with Adversarial Examples“, 24. Februar 2017. <https://openai.com/blog/adversarial-example-research/>.
- Azulay, Aharon, und Yair Weiss. „Why Do Deep Convolutional Networks Generalize so Poorly to Small Image Transformations?“, o. J., 25.
- Ballard, Dana Harry, und Christopher M. Brown. *Computer Vision*. Prentice-Hall, 1982.
- Bhagoji, Arjun Nitin, Warren He, Bo Li, und Dawn Song. „Practical Black-Box Attacks on Deep Neural Networks Using Efficient Query Mechanisms“. In *Computer Vision – ECCV 2018*, herausgegeben von Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, und Yair Weiss, 11216:158–74. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018. [https://doi.org/10.1007/978-3-030-01258-8\\_10](https://doi.org/10.1007/978-3-030-01258-8_10).
- Brynjolfsson, und McAfee. „The Business of Artificial Intelligence“, o. J., 20.
- Buchkremer, Rüdiger, Alexander Demund, Stefan Ebener, Fabian Gampfer, David Jägering, Andreas Jürgens, Sebastian Klenke, u. a. „The Application of Artificial Intelligence Technologies as a Substitute for Reading and to Support and Enhance the Authoring of Scientific Review Articles“. *IEEE Access* 7 (2019): 65263–76. <https://doi.org/10.1109/ACCESS.2019.2917719>.
- Canziani, Alfredo, Adam Paszke, und Eugenio Culurciello. „An Analysis of Deep Neural Network Models for Practical Applications“. *ArXiv:1605.07678 [Cs]*, 14. April 2017. <http://arxiv.org/abs/1605.07678>.
- Carlini, Nicholas, und David Wagner. „Towards Evaluating the Robustness of Neural Networks“. *arXiv:1608.04644 [cs]*, 22. März 2017. <http://arxiv.org/abs/1608.04644>.
- „Computer Vision - Dana H. Ballard, Christopher M. Brown - Google Books“. Zugegriffen 1. August 2020. <https://books.google.de/books?id=xZ0ayAEACAAJ&dq=Computer+Vision++Dana+H.+Ballard&hl=en&sa=X&ved=2ahUKEwiEkIOR5PnqAhWF5KQKHdAjDCoQ6AEwAHoECAAQAQ>.
- Verified Market Research. „Computer Vision Market Size, Share, Industry Analysis, Trends & Forecast“. Zugegriffen 10. August 2020. <https://www.verifiedmarketresearch.com/product/global-computer-vision-market-size-and-forecast-to-2025/>.
- „Computer Vision Technology in Agricultural Automation - A Review“. Zugegriffen 10. August 2020. <https://doi.org/10.1016/j.inpa.2019.09.006>.
- Das, Nilaksh, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E. Kounavis, und Duen Horng Chau. „Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression“. *arXiv:1705.02900 [cs]*, 8. Mai 2017. <http://arxiv.org/abs/1705.02900>.
- Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, und Li Fei-Fei. „ImageNet: A

- large-scale hierarchical image database“. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–55, 2009. <https://doi.org/10.1109/CVPR.2009.5206848>.
- DeVries, Terrance, und Graham W. Taylor. „Improved Regularization of Convolutional Neural Networks with Cutout“. *arXiv:1708.04552 [cs]*, 29. November 2017. <http://arxiv.org/abs/1708.04552>.
- Dodge, Samuel, und Lina Karam. „Understanding how image quality affects deep neural networks“. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, 1–6, 2016. <https://doi.org/10.1109/QoMEX.2016.7498955>.
- Elsayed, Gamaleldin, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, und Jascha Sohl-Dickstein. „Adversarial Examples that Fool both Computer Vision and Time-Limited Humans“. In *Advances in Neural Information Processing Systems 31*, herausgegeben von S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, und R. Garnett, 3910–3920. Curran Associates, Inc., 2018. <http://papers.nips.cc/paper/7647-adversarial-examples-that-fool-both-computer-vision-and-time-limited-humans.pdf>.
- Foley, David, und Ruairi O’Reilly. „An Evaluation of Convolutional Neural Network Models for Object Detection in Images on Low-End Devices“, o. J., 12.
- „Fooling Google’s image-recognition AI 1000x faster | MIT CSAIL“. Zugegriffen 16. August 2020. <https://www.csail.mit.edu/news/fooling-googles-image-recognition-ai-1000x-faster>.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, und Jian Sun. „Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification“. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–34. Santiago, Chile: IEEE, 2015. <https://doi.org/10.1109/ICCV.2015.123>.
- Hosseini, Hossein, Baicen Xiao, und Radha Poovendran. „Google’s Cloud Vision API Is Not Robust To Noise“. *arXiv:1704.05051 [cs]*, 20. Juli 2017. <http://arxiv.org/abs/1704.05051>.
- Iandola, Forrest N., Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, und Kurt Keutzer. „SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size“. *arXiv:1602.07360 [cs]*, 4. November 2016. <http://arxiv.org/abs/1602.07360>.
- Ilyas, Andrew, Logan Engstrom, Anish Athalye, und Jessy Lin. „Black-box Adversarial Attacks with Limited Queries and Information“. *arXiv:1804.08598 [cs, stat]*, 11. Juli 2018. <http://arxiv.org/abs/1804.08598>.
- Ilyas, Andrew, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, und Aleksander Madry. „Adversarial Examples Are Not Bugs, They Are Features“. *arXiv:1905.02175 [cs, stat]*, 12. August 2019. <http://arxiv.org/abs/1905.02175>.
- Juuti, Mika, Buse Gul Atli, und N. Asokan. „Making targeted black-box evasion attacks effective and efficient“. *arXiv:1906.03397 [cs, stat]*, 8. Juni 2019. <http://arxiv.org/abs/1906.03397>.
- Kouzis-Loukas, Dimitrios. *Learning Scrapy*. Packt Publishing Ltd, 2016.
- Koziarski, Michał, und Bogusław Cyganek. „Image Recognition with Deep Neural Networks in Presence of Noise – Dealing with and Taking Advantage of Distortions“. *Integrated Computer-Aided Engineering* 24, Nr. 4 (5. September 2017): 337–49. <https://doi.org/10.3233/ICA-170551>.
- Krizhevsky, Alex, Ilya Sutskever, und Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In *Advances in Neural Information Processing Systems 25*, herausgegeben von F. Pereira, C. J. C. Burges, L. Bottou,

- und K. Q. Weinberger, 1097–1105. Curran Associates, Inc., 2012. <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- Li, Xurong, Shouling Ji, Meng Han, Juntao Ji, Zhenyu Ren, Yushan Liu, und Chunming Wu. „Adversarial Examples Versus Cloud-based Detectors: A Black-box Empirical Study“. *arXiv:1901.01223 [cs]*, 14. September 2019. <http://arxiv.org/abs/1901.01223>.
- Lu, Jiajun, Hussein Sibai, Evan Fabry, und David Forsyth. „NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles“. *arXiv:1707.03501 [cs]*, 11. Juli 2017. <http://arxiv.org/abs/1707.03501>.
- Mahony, Niall O’, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco-Hernandez, Lenka Krpalkova, Daniel Riordan, und Joseph Walsh. „Deep Learning vs. Traditional Computer Vision“. *arXiv:1910.13796 [cs]* 943 (2020). <https://doi.org/10.1007/978-3-030-17795-9>.
- Mitchell, Ryan. *Web Scraping with Python: Collecting More Data from the Modern Web*. O’Reilly Media, Inc., 2018.
- Mohanta, Raj Kumar, und Binapani Sethi. „AMAZON REKOGNITION FOR PATTERN RECOGNITION“ 11, Nr. 6 (o. J.): 4.
- Olston, Christopher, und Marc Najork. *Web Crawling*. Now Publishers Inc, 2010.
- Prince, Simon J.D. „Computer Vision: Models, Learning, and Inference - Simon J. D. Prince - Google Books“. Zugegriffen 19. Juli 2020. [https://books.google.de/books?id=PmrICLzHutgC&printsec=frontcover&dq=Computer+Vision:+Models,+Learning,+and+Inference&hl=en&sa=X&ved=2ahUKewj\\_x6qO-9jqAhWP\\_qQKHXYX1AzUQ6AEwAHoECAEQAg#v=snippet&q=Page%20ix&f=false](https://books.google.de/books?id=PmrICLzHutgC&printsec=frontcover&dq=Computer+Vision:+Models,+Learning,+and+Inference&hl=en&sa=X&ved=2ahUKewj_x6qO-9jqAhWP_qQKHXYX1AzUQ6AEwAHoECAEQAg#v=snippet&q=Page%20ix&f=false).
- Pritt, Mark, und Gary Chern. „Satellite Image Classification with Deep Learning“. In *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 1–7. Washington, DC, USA: IEEE, 2017. <https://doi.org/10.1109/AIPR.2017.8457969>.
- Román-Palacios, Cristian, und John J. Wiens. „Recent Responses to Climate Change Reveal the Drivers of Species Extinction and Survival“. *Proceedings of the National Academy of Sciences* 117, Nr. 8 (25. Februar 2020): 4211–17. <https://doi.org/10.1073/pnas.1913007117>.
- Shapiro, Linda G., und George C. Stockman. *Computer Vision*. Prentice Hall, 2001.
- Stutz, David, Matthias Hein, und Bernt Schiele. „Disentangling Adversarial Robustness and Generalization“. *arXiv:1812.00740 [cs, stat]*, 10. April 2019. <http://arxiv.org/abs/1812.00740>.
- Temel, Dogancan, Gukyeong Kwon, Mohit Prabhushankar, und Ghassan AlRegib. „CURE-TSR: Challenging Unreal and Real Environments for Traffic Sign Recognition“, 12. Oktober 2017. <https://openreview.net/forum?id=Hy4q48h3Z>.
- Tollefson, Jeff. „One Million Species Face Extinction“, o. J., 1.
- Trucco, Emanuele, und Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- Walt, Stefan van der, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, und the scikit-image contributors. „scikit-image: Image processing in Python“. *PeerJ* 2 (19. Juni 2014): e453. <https://doi.org/10.7717/peerj.453>.
- Wu, Ren, Shengen Yan, Yi Shan, Qingqing Dang, und Gang Sun. „Deep Image: Scaling up Image Recognition“. *arXiv:1501.02876 [cs]*, 5. Juli 2015.

<http://arxiv.org/abs/1501.02876>.

- Xie, Qizhe, Minh-Thang Luong, Eduard Hovy, und Quoc V Le. „Self-Training With Noisy Student Improves ImageNet Classification“, o. J., 12.
- Xin, Feng, Jiang Youni, Yang Xuejiao, Du Ming, und Li Xin. „Computer Vision Algorithms and Hardware Implementations: A Survey | Elsevier Enhanced Reader“. Zugegriffen 19. Juli 2020. <https://doi.org/10.1016/j.vlsi.2019.07.005>.
- Zhou, Brady, Philipp Krähenbühl, und Vladlen Koltun. „Does computer vision matter for action?“ *Science Robotics* 4, Nr. 30 (22. Mai 2019): eaaw6661. <https://doi.org/10.1126/scirobotics.aaw6661>.
- Zhou, Yiren, Sibong Song, und Ngai-Man Cheung. „On Classification of Distorted Images with Deep Convolutional Neural Networks“. *arXiv:1701.01924 [cs]*, 8. Januar 2017. <http://arxiv.org/abs/1701.01924>.

## Internetquellen

- Helium Software. „Web Scraper | Helium Scraper | Buy“, 25. August 2020. <https://www.heliumscraper.com/eng/buy.php>.
- Helium Software. „Web Scraper: Best Web Scraping Tool to Extract Data from Websites“, 25. August 2020. <https://www.heliumscraper.com/eng/>.
- Marr, Bernard. „3 (Not So Obvious) Industries That Could Be Transformed By Computer Vision“. *Forbes*. Zugegriffen 10. August 2020. <https://www.forbes.com/sites/bernardmarr/2020/06/12/3-not-so-obvious-industries-that-could-be-transformed-by-computer-vision/>.
- OutWit Technologies. „Harvest the Web | OutWit“, 25. August 2020. <https://www.outwit.com/#hub>.
- Sequentum. „Overview | Data Extraction Software | Visual Web Ripper“, 25. August 2020. <http://visualwebripper.com/Product>.
- Sitebulb Limited. „How to Crawl Responsibly: The Need for (Less) Speed“. Sitebulb, 25. August 2020. <https://sitebulb.com/resources/guides/how-to-crawl-responsibly-the-need-for-less-speed/>.



## Ehrenwörtliche Erklärung

Hiermit versichern wir, dass die vorliegende Arbeit von uns selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass wir alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet haben. Wir versichern auch, dass die von uns eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erklären wir, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Wir erklären uns damit nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Wir erklären uns damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Köln, 26.08.2020



Neli Garkova



Dominik Mazurkiewicz



Simon Schell