Wally Image Processing Model
AutoML Vision vs. Handmade Model
AVPRG Projektkonzept

Taras Sologub | Mat. Nr. 2153832
Tom Klanthe | Mat. Nr. 2285567

Einleitung

Unser Projekt hat den Titel “Wally Image Processing Model”. Das übergeordnete Ziel dieses Projektes ist das Erlernen und Anwenden von Machine Learning Technologien. Die Lehrveranstaltung eignet sich dazu besonders gut, da wir mittels OpenCV Bilddaten erheben können, welche wir dann in unseren Machine Learning Tools benutzen. Eine weitere Motivation dieses Projektes ist der Vergleich zwischen verschiedenen Machine Learning Technologien.
Wenn das Projekt fertiggestellt ist, werden wir es in Form einer normalen Folienpräsentation und Vorführung des Endergebnisses vorstellen. Die Präsentation wird unsere Herangehensweise, benutzte Technologien sowie Probleme und eventuelle Hindernisse, auf die wir im Verlauf des Projektes gestoßen sind, erläutern.

Projektziel

1987 erschien erstmals die Kinderbuchreihe “Where’s Wally?” in Großbritannien. In den Büchern befinden sich sehr große und detailreiche Bilder, welche verschiedene Schauplätze darstellen. Diese sind mit sehr vielen Charakteren gefüllt. Ziel ist es einen Charakter, “Wally”, zu finden. Er hat immer das gleiche Aussehen, ist jedoch wegen dem Detailreichtum der Bilder schwer zu finden.
In unserem Projekt wollen wir eine Software entwickeln, welche es ermöglicht Wally mit Hilfe von Machine Learning in nur wenigen Sekunden zu finden.
Der erste Schritt ist das Filmen einer Buchseite mit einer Kamera. Die Kamera wird auf einem Stativ befestigt und zeichnet eine gesamte Seite auf.
Die Software wird gestartet und die Bilddaten werden dann mit OpenCV gesammelt. Dann wird Machine Learning benutzt um in den gewonnenen Bilddaten Wally zu finden.
Das gefilmte Bild wird auf dem Rechner, auf dem auch die Software läuft, angezeigt. Wenn Wally gefunden wurde, wird er auf dem Bild markiert bzw. grafisch hervorgehoben.
Wir werden auch versuchen die Software dauerhaft laufen zu lassen, sodass Wally live gefunden werden kann, wenn man zwischendurch auf eine andere Seite umblättert.
Dieser Ablauf wird auch so in der Vorführung zu sehen sein.

Ein weiteres Projektziel ist der Vergleich zwischen zwei Machine Learning Technologien. Eine werden wir selber mit Hilfe von Python programmieren, die andere wird von Google zur Verfügung gestellt und nennt sich Cloud ML Engine.

Anforderungsanalyse

Das Hauptziel unserer Anwendung ist es mit einer hohen Genauigkeit und geringer Rechenzeit ein gegebenes Pattern in einem unbekannten Bild zu finden (mit anderen Worten Wally finden). Im Vorfeld muss die Software mit Wally Bildern “gefüttert” werden, damit sie während des Suchvorgangs eine Referenz hat.
Das Projekt ist nicht Usergebunden, da es in der realen Welt keinen konkreten Anwendungsfall gibt. Es soll nur dazu dienen moderne ML Technologien kennen zu lernen und durch das Projekt zu veranschaulichen.

Das Projekt gilt für uns als erfolgreich, wenn wir am Ende eine Genauigkeit von mindestens 95% und eine Rechenzeit von <5 Sekunden erzielen.
Wir wollen auch versuchen die Software flexibel zu gestalten, damit sie auf umblättern der Buchseiten, Bewegung der Kamera und unterschiedliche Blickwinkel reagieren kann.
Ein weiteres Ziel ist auch der Vergleich zweier Machine Learning Technologien. Wir werden eine eigens implementierte Python Lösung mit einer von Google bereitgestellten Lösung vergleichen und analysieren in welchen Bereichen es Vor- und Nachteile gibt.

Technische Rahmenbedingungen.

In unserem Projekt geht es hauptsächlich um Machine Learning und Bildanalyse, das heißt Python wird unser Hauptwerkzeug sein. Python ist heutzutage die Hauptprogrammiersprache, wenn es um Machine Learning oder Deep Learning Entwicklung geht. Alle Frameworks, APIs und Bibliotheken, die wir verwenden werden funktionieren in Python.

In unserem Projekt verwenden wir hauptsächlich folgende Komponenten:

- **Python** – Python ist eine dynamische Programmiersprache, welche es ermöglicht einen fertigen Prototyp in kurzer Zeit zu kreieren. Nachdem man die Grundlagen verstanden hat, ist die Lernkurve sehr steil und man kann sofort mit der Realisierung anfangen und Konzepte schnell validieren. Python hat auch die größte Menge von Machine Learning Frameworks und Bibliotheken.

- **Hard- und Softwareumgebung** – Wichtig ist die Python Unterstützung. Wir werden unser Programm auf einem Windows und Mac benutzen, andere Umgebungen funktionieren jedoch auch

- **Kamera** - Wir benutzen eine Logitech C270 Webcam – Sie hat eine hohe Auflösung, damit wir die detailreichen Bilder in möglichst guter Qualität aufnehmen können.

- **Google AutoML Vision** – ein Service von Google. Dieser bietet sowohl vortrainierte Modelle über eine API als auch die Möglichkeit, kundenspezifische Modelle mit AutoML Vision zu erstellen. Die Cloud Vision API ermöglicht den Inhalt eines Bildes zu verstehen, indem sie leistungsstarke maschinelle Lernmodelle in einer einfach zu bedienenden REST-API kapselt. Es klassifiziert Bilder schnell in Kategorien und erkennt einzelne Objekte und Gesichter in Bildern. Nach dem Hochladen und Beschriften von Bildern trainiert AutoML Vision ein Modell.

- **OpenCV mit Python API** – Wird benötigt um unsere eigene Machine Learning Lösung zu implementieren. Die Software nimmt Bilder über die Kamera auf und findet Gesichter auf dem Bild, welche dann weiter verarbeitet werden können.

- **TensorFlow** – ist eine Open-Source-Bibliothek für numerische Berechnungen und groß angelegtes maschinelles Lernen. TensorFlow bündelt eine Reihe von Modellen und Algorithmen für Machine Learning und Deep Learning. Es wird Python verwendet, um eine komfortable Front-End-API für die Erstellung von Anwendungen mit dem Framework

bereitzustellen, während diese Anwendungen in C++ ausgeführt werden. Mit TensorFlow können die Entwickler Dataflow-Grafiken erstellen womit man sehen kann, wie sich Daten durch eine Reihe von Verarbeitungsknoten bewegen. Jeder Knoten im Diagramm stellt eine mathematische Operation dar, und jede Kante zwischen den Knoten ist ein multidimensionales Datenarray oder ein Tensor.
Das Mobilenet-Modell wurde durch ein ImageNet-Dataset vortrainiert. Wir erstellen eine neue oberste Ebene, welche Bilder einer anderen Art erkennt.

- **Haar Cascade Classifier Algorithm** – ein Algorithmus der entscheidet, ob ein Bild ein positiv (Gesicht vorhanden) oder ein negativ (kein Gesicht vorhanden) ist. Ein Klassifikator wird auf Hunderttausenden von Gesichts- und Nicht-Gesichtsbildern geschult, um zu lernen, wie man ein neues Bild richtig klassifiziert. OpenCV bietet zwei vortrainierte Klassifikator für die Gesichtserkennung: Den Haar Klassifikator und den LBP Klassifikator. Beide Klassifikator verarbeiten Bilder in Graustufen, da man keine Farbinformationen benötigt, um zu entscheiden, ob ein Bild ein Gesicht beinhaltet oder nicht. „Vortrainiert" bedeutet, dass die Klassifikator auf eine Datei zugreifen, welche bereits Daten für die Erkennung von Gesichtern. Um einen Klassifikator auszuführen, müssen wir zuerst diese Datei laden.
  Beide OpenCV Gesichtserkennungklassifikatoren haben ihre Vor- und Nachteile, die größten Unterschiede liegen jedoch in der Genauigkeit und Geschwindigkeit.

Algorithmus	Vorteile	Nachteile
Haar Klassifikator	1. Hohe Erkennungsgenauigkeit. 2. Niedrige False-Positive Rate.	1. Komplexe Berechnung. 2. Längere Trainingszeit. 3. Weniger genau auf schwarzen Flächen. 4. Einschränkungen bei schwierigen Beleuchtungsverhältnissen. 5. Weniger Robust gegenüber Okklusion.
LBP Klassifikator	1. Schnelle und einfache Berechnung. 2. Kürzere Trainingszeit. 3. Robust gegenüber lokalen Beleuchtungsänderungen. 4. Robust gegenüber Okklusion.	1. Geringe Genauigkeit. 2. Hohe False-Positive-Rate.

Wenn also genauere Erkennungen erforderlich sind, ist der Haar-Klassifikator der richtige Weg. Die Genauigkeit war für uns ein kritischer Entscheidungspunkt.

Technisches Konzept

Wir wollen zwei Bildanalyse Realisierungen entwickeln um Wally zu finden und die Resultate vergleichen. Am wichtigsten ist für uns die Genauigkeit und Geschwindigkeit. Bei der ersten Realisierung handelt es sich um unsere eigene Lösung. Sie basiert auf dem vortrainierten TensorFlow Modell „Mobilenet“. Die zweite Realisierung benutzt auch ein vortrainiertes Modell, aber von Google Cloud Vision.
Bevor Machine Learning angewandt werden kann brauchen wir ein möglichst großes Dataset mit Wally Bildern. Die Bilder werden wir manuell mit Hilfe diverser Suchmaschinen sammeln.
Wenn das Dataset groß genug ist kann es für unser Deep Learning Modell verwendet werden.

Der genaue Ablauf des Programms sieht wie folgt aus:

- Scannen und speichern des Bildes aus dem Buch mit der OpenCV API
- Parsen des Bildes mit Hilfe von Haar Cascade Algorithmus (Gesichter werden erkannt, extrahiert und dann gespeichert)
- Die Gesicht Objekte werden an Auto ML Vision und an unsere eigene Implementierung weitergegeben.
- Die Objekte werden durch die Modelle analysiert.
- Das Objekt welches Wally am ähnlichsten sieht wird uns zurückgegeben
- Mit Hilfe der Koordinaten des Objektes markieren wir Wally auf dem von uns ursprünglich aufgenommenen Bild

Bedienkonzept

Das Projekt ist nicht Usergebunden, da es in der realen Welt keinen konkreten Anwendungsfall gibt. Es soll nur dazu dienen moderne ML Technologien kennen zu lernen und durch das Projekt zu veranschaulichen. D.h. es gibt kein wirkliches Bedienkonzept.

Zeitplan

Unsere Gruppe besteht aus zwei Mitglieder, deswegen planen wir circa 10 Stunden/Person pro Woche Zeit zu investieren. Das ergibt sich 6 Wochen * 10 = 60 Personenstunden. Wir starten am 31.Oktober mit der Deadline für den ersten Meilenstein (26. November) und mit dem Enddatum für den zweiten Meilenstein (15.Dezember).

Die Hauptziele bis zum ersten Meilenstein (26. November):

- erlernen der Grundlagen von Machine Learning und Deep Learning (Stichworte und Begrifflichkeiten)
- was ist das Google AutoML Vision? Welche API hat er? Wie kommuniziert der Enduser mit diesem Service?
- Input Daten sammeln für Deep Learning
- entscheiden welchen Machine Learning Lernstil (supervised, unsupervised oder reinforcemen) wir als Basis nehmen.
- welche OpenCV Algorithmen existieren? Welche funktionieren am besten für Bildanalyse?

- nachtrainierbare Modelle vs. vortrainierte Modelle. Welche passen am besten für unseren Anwendungsfall?

Das Ziel dieses Meilensteins ist es zwei fertige ML Modelle (Auto ML Vision und eigene Implementierung) zu haben. Diese sollen erst einmal auf ein Standbild trainiert werden und mit schlechter Präzision Wally auf dem Bild finden. Falls wir das erreichen, kann man diesen Meilenstein als erfolgreich ansehen.

Die Hauptziele für den 2. Meilenstein (15. Dezember):
- Das Bild von einer Webcam mit OpenCV auslesen und parsen.
- Die Modelle auf mehreren Bildern (aus dem Buch) anwenden.
- die Präzision verbessern.

Das Ziel dieses Meilensteins ist es ein fertiges Produkt entwickelt zu haben. D.h. zwei funktionierende Modelle implementiert haben, die Wally mit einer Genauigkeit von mindestens 90% finden und die Berechnung soll unter 5 Sekunden.

Teamplanung
Das Team besteht aus zwei Personen: Taras Sologub und Tom Klanthe.
Da wir zwei Machine Learning Technologien vergleichen wollen teilen wir diese auch auf uns beide auf. Taras kümmert sich um die eigene Implementierung in Python, Tom kümmert sich um die Google ML Cloud Lösung.
Wir werden uns beide unabhängig voneinander mit ML beschäftigen und am Projektende die Differenzen der beiden Technologien analysieren. Selbstverständlich helfen wir uns gegenseitig weiter, sollte einer an ein Hindernis geraten.