Erweiterung eines Trainingdatensatzes für einen Objektdetektor im Kontext eines botanischen Gartens mit Methoden der Data Augmentation

Daniel Eggert

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG,
Berliner Tor 7, 20099 Hamburg

Zusammenfassung. Um eine zuverlässige Objekterkennung oder Objektdetektion zu gewährleisten, führt kein Weg an guten sowie ausreichenden Trainingsdaten vorbei. Doch sind ausreichend viele Trainingsdaten nicht immer vorhanden. Die vorliegende Arbeit beschäftigt sich deshalb mit der Erweiterung von Original-Daten, um damit das Training eines Objektdetektors zu verbessern. Der Objektdetektor soll den Grundbaustein einer AR-Anwendung bilden, welche als Guide für die Tour durch einen botanischen Garten dienen soll.

1 Einleitung

Ziel der zu entwickelnden Anwendung ist es, verschiedene Objekte im botantischen Garten Klein-Flottbek in Hamburg mit dem Handy über eine AR Umgebung zu erkennen und Informationen zu diesen anzuzeigen. Bei den Objekten handelt es sich um Pflanzen, Tiere, Skulpturen und Gebäude. Das größte Problem stellt hier die Beschaffung der Trainingsdaten dar.

Das Problem von zu wenigen Daten ist ein häufig auftretendes Bottleneck beim Training von neuronalen Netzen.[13]

Die Problemstellung bietet den Vorteil, dass die Grundwahrheit(engl. Ground Truth) zu den Objekten durch Schilder bereitgestellt wird. Des weiteren sind auch die Orte an denen die Objekte erkannt werden müssen fest. Trotzdem bestehen weitere Herausforderungen an die Trainingsdaten, welche in dem folgenden Kapitel näher spezifiziert werden. Anschließend wird ein Überblick der Lösungsmöglichkeiten für die Erweiterung eines bereits bestehenden Datensatzes geschaffen. Anhand der entdeckten Lösungsmöglichkeiten wird ein Konzept erstellt, um den bereits bestehenden Datensatz zu erweitern. Darauf folgend wird die praktische Umsetzung des Konzepts aufgezeigt. Abschließend werden in einem Fazit die Ergebnisse prägnant zusammengefasst, die Probleme, welche durch augmentierte Daten bezüglich eines Objektdetektors entstehen, kritisch hervorgehoben und die nächsten möglichen Schritte aufgezeigt.

2 Herausforderungen der Anwendungsumgebung

2.1 Wetterbedingungen

Das Sammeln von Trainingsdaten zu jeden Wetterbedinungen ist schwierig. Jedoch beeinflusst das Wetter auch die Erkennung von Objekten. So handelt es sich bei Wettereffekten um nur schwer erfassbare Daten, welche jedoch einen großen Einfluss auf die Robustheit von neuronalen Netzen haben. Gerade deshalb ist es wichtig, diese Umgebungseffekte auch schon beim Training zu berücksichtigen. Viele existente Datensätze berücksichtigen jedoch nur Bilder unter guten Wetterbedingungen.[17]

2.2 Veränderlichkeit der Objekte

Um eine App zu entwickeln, welche auch mehrere Pflanzen erkennen soll, muss die Pflanze in jeglicher Form erkannt werden. Da sich die meisten Pflanzen über die Jahreszeiten verändern, müssten Trainingsdaten der Pflanzen zu jeder Jahreszeit gesammelt werden. Dies ist aber nicht möglich, wenn man nicht die Zeit hat ein Jahr zu warten. Auch kann sich die Form einer Pflanze z.B. durch handwerkliche Arbeiten an dieser verändern. Dies ist beispielsweise der Fall, wenn diese zurückgeschnitten wird.

2.3 Objektabdeckung

Das in der Realität Objekte ganz oder teilweise durch andere Objekte abgedeckt sind, ist ein in natürlichen Umgebungen häufig auftretendes Ereignis. Objektabdeckungen stellen eine große Herausforderung für die Erkennung und Detektion von Objekten dar, denn diese reagieren sehr empfindlich auf solche Abdeckungen. Auch können diese Probleme nicht immer durch ein Training mit teilweise abgedeckten Bildern behoben werden.[18]

Dies ist auch in Stadtparks der Fall, denn je nach Anzahl von Besuchern ist ein Objekt besser oder schlechter zu sehen. Manchmal können andere Besucher die Objekte ganz oder zumindest teilweise verdecken oder Schatten auf die Objekte werfen. Ein solcher Zustand ist nur schwer real aufzunehmen. Bei Tieren könnte ein Problem sein, dass diese nur teilweise zu sehen sind, weil sie sich gerade hinter anderen Objekten verstecken.

3 Verwandte Arbeiten und Lösungsmöglichkeiten

In der Literatur sind drei Lösungsmöglichkeiten für die Herausforderung zu weniger Daten ersichtlich. Eine ist die Suche nach Daten mit der gleichen Domain z.B. im Internet. Hier kann man auch einen Objekterkenner benutzen, um die Bilder herauszufiltern, welche das gesuchte Objekt enthalten. Dies ist vor allem nützlich, wenn die Bilder nicht beschriftet sind und Sammlungen von Bildern untersucht werden sollen.[6]

Synthetische Daten sind die andere Möglichkeit das Problem von zu wenigen Daten zu lösen. Mit Hilfe synthetischer Daten kann man ethische wie z.B. datenschutzrechtliche oder praktische Herausforderungen umgehen. Die besten Effekte mit synthetischen Daten werden erzielt, wenn man sie mit realen Daten mischt.[13][6]

Data Augmentation ist die dritte Möglichkeit den Datensatz zu erweitern. Dabei werden Original-Bilder mit den Funktionen der Bildbearbeitung verändert[14]. Zusätzlich kann die Data Augmentation auch auf synthetische Daten angewandt werden[6].

3.1 GAN's

Bei GAN'S(Generative Adversarial Networks) handelt es sich um neuronale Netze, die erlernen, synthetische Bilder zu generieren, welche denen der Trainingsdaten sehr ähnlich sind. Ein Beispiel dafür sind Gesichter von Menschen. Anhand vieler Gesichter als Trainingsdaten kann ein GAN Bilder von Gesichtern generieren, welche es so auf der Welt gar nicht gibt. Das Training eines GAN's geschieht mit Hilfe eines Generators und Diskriminators. Das GAN ist der Generator und ein Klassifizierer ist der Diskriminator. Während der Generator Bilder erstellt, versucht der Diskriminator zu erkennen, ob es sich um ein Original-Bild handelt oder um eine Fälschung. Der Generator versucht in diesem Verfahren folglich so gute Fälschungen zu erstellen, dass diese nicht mehr von Originalen unterschieden werden können.[13]

Allerdings wurden die Ergebnisse von GANs früher als nicht realistisch genug angesehen, um die Daten für eine signifikante Verbesserung des Trainings eines CNNs zu verwenden. Die Realistik von GANs hat sich jedoch im Laufe der Jahre verbessert. Dennoch existieren noch Probleme bei der Verwendung von GANs zur Erzeugung synthetischer Daten. Ein Beispiel hierfür ist der mögliche Verlust der Grundwahrheit der markierten Objekte. Wenn sich die markierten Objekte auf den künstlichen Bildern zu sehr von den realen unterscheiden und für das Training verwendet werden, können diese die Performance eines neuronalen Netzes sogar vermindern. Auch führt eine zu sehr auf Zufall basierende Ergänzung eines Datensatzes mit synthetischen Daten nicht immer zu einer gesteigerten Performance der Erkennung. Deshalb sollten nicht alle Ergebnisse von GANs für das Training verwendet werden, sondern nur die Ergebnisse, welche der Realität sehr nahe kommen. Dafür kann z.B. auch ein weiteres CNN verwendet werden, welches mit einem geringeren Datensatz darauf trainiert wurde, die Objekte zu erkennen, welche ein GAN noch erhalten soll.[3]

3.2 Data Augmentation

Data Augmentation ist eine Möglichkeit, einen bestehenden Trainingsdatensatz zu erweitern, um damit z.B. Overfitting zu minimieren. Ziel der Data Augmentation ist es, die Probleme der Objekterkennung wie beispielsweise die Objektabdeckung, die Objektbeleuchtung sowie der Bildhintergründe zu lösen.[14] Ein Beispiel für den Effekt von Bildhintergründen für das Training wäre es, wenn

das neuronale Netz lernt, eine Möwe daran zu erkennen, dass sich der Sand vom Strand im Hintergrund befindet[12]. Bei der Data Augmentation wird zwischen der offline und der online Data Augmentation unterschieden. Die offline Data Augmentation wird vor dem Training eines Models ausgeführt und erweitert den Trainingsdatensatz. Die online Data Augmentation findet während des Trainings innerhalb des zu trainierenden Models statt und die augmentierten Daten werden auch nicht gespeichert, sondern ausschließlich für das Training verwendet. Dadurch wird beispielsweise Speicher auf der Festplatte im Gegensatz zu der offline Data Augmentation eingespart.[15]

Für diese Arbeit wird sich jedoch nur mit der offline Data Augmentation beschäftigt, denn es ist das Ziel den Trainingsdatensatz zu erweitern, da in Create ML[1] das Training nur eingeschränkt beeinflusst werden kann.

Die offline Data Augmentation führt einfache Datenverzerrungen aus. Dabei wird z.B. ein Bild gespiegelt, verpixelt oder rotiert[15]. Um einem Bild künstlich Wettereffekte hinzuzufügen können beispielsweise auch Regensimulatoren zum Einsatz kommen.[17]

Für kleinere Änderungen genügen Bibliotheken wie OpenCV und PIL. Diese Bibliotheken können genutzt werden um unter anderem Kanten zu verstärken oder Objekte zu glätten, oder Formen und Farben einzuzeichnen. Mit der PIL Bibliothek können auch komplexere Strukturen in Bilder eingefügt werden.[11] Thi Huyen Cao verwendete in ihrer Arbeit den von Keras bereitgestellten ImageDataGenerator, um damit ihre Trainingsdaten offline sowie online zu augmentieren.[5]

3.3 Rendering-Pipeline

Eine Möglichkeit synthetische Bilder zu generieren, stellen Borkman et al vor. Dabei handelt es sich um Unity Perception. Unity Perception ist eine Open Source Bibliothek, welche Werkzeuge enthält, um aus künstlich generierten Szenen in Unity gekennzeichnete Trainingsbilder zu erstellen. Das Paket bietet kontrollier- und anpassbare Umgebungen, welche 3D-Assets beinhalten. Die 3D Assets sind im Projekt beschriftet. Diese Beschriftung ist die Grundwahrheit der Objekte, welche im Bild gelabelt und per Rechteck markiert werden. Zusätzlich zu den bereits genannten Optionen bietet Unity Perception die Möglichkeit, mit Zufälligkeitsparametern Variationen hervorzurufen. Ein Beispiel für diese Parameter ist z.B. die Beleuchtung der Szene. Auch Objekte im Hintergrund oder Vordergrund können mit diesen Zufälligkeitsparametern eingestellt werden. Interessant ist auch die Nachbereitung der Kamera, so kann der Kontrast und die Sättigung zufällig verändert werden. Die Autoren haben mit Unity Perception 400000 synthetische Trainingsbilder generiert. Sie konnten feststellen, dass diese mit einer Mischung mit realen Daten die Performance von neuronalen Netzen verglichen mit dem selben Netz, welches nur auf reale Daten trainiert wurde, gesteigert hat.[4]

4 Konzept

Diese Arbeit erweitert die Trainingspipeline des Setups von Daniel Eggert[7]. Darin wurde bereits ein Setup entwickelt, welches Boundingboxen mit Beschriftungen anzeigt. Das Setup wurde zunächst für die Unterscheidung von Obst- und Gemüsesorten entwickelt. Der Use-Case lässt sich allerdings ohne größeren Aufwand verändern, da die angezeigten Informationen lediglich vom verwendeten neuronalen Netz abhängig sind. Lediglich das neuronale Netz muss über Create ML neu trainiert werden.

Der Fokus dieser Arbeit liegt nicht auf einer Erweiterung des Setups, sondern einer Verbesserung der Trainingspipeline. Create ML[1] lässt wenig Änderungen an den Parametern des Modells zu, deshalb sind hier die Trainingsdaten um so wichtiger.

Zunächst wird ausschließlich die Methode der Data Augmentation verwendet. Dies begründet sich aus der Tatsache, dass die Generierung synthetischer Daten mit einem GAN von der Performance eines Objekterkenners abhängig ist. Der Objekterkenner soll schließlich die synthtischen Daten auf ihre Ground Truth überprüfen. Mit dem augmentierten Datensatz könnte anschließend auch der für den GAN benötige Objekterkenner trainiert werden. Auch auf den Aufbau einer Rendering Pipeline wird zunächst verzichtet, da diese im Anschluss auf die augmentierten Daten angewendet werden könnte. Würde die Rendering Pipeline zuerst implementiert werden, würden auch die von dieser generierten Objekte verändert werden.

Die Technologien, die für diese Aufgabe verwendet werden, sind OpenCV[16], Pillow[8] und Keras[9]. Anhand implementierter Funktionen der genannten Bibliotheken sollen bereits vorhandene Trainingsdaten in einem Pythonskript entgegen genommen werden. Diese Bilder werden dann mittels der Methoden verändert und der Trainingsdatensatz wird durch augmentierte Daten erweitert.

Ziel ist es, mindestens 25 Bilder zu generieren. Der Dokumentation[2] von Create ML ist zu entnehmen, dass pro Objekt mindestens 25 Bilder vorhanden sein sollen. Dies wäre dann schon bei einem Original-Bild der Fall, da durch ein Bild bereits 25 Bilder zur Verfügung stehen würden. Wichtig ist auch, dass nicht mehr Bilder eines Objektes als von einem anderen Objekt für das Training verwendet werden. Die Bilder sollen folglich gleichermaßen verteilt sein. Deshalb ist es wichtig, dass jede Funktion auch für jedes verwendete Bild eingesetzt wird.

5 Praktische Umsetzung

Das Programm für die Erweiterung des Trainingdatensatzes mittels Data Augmentation besteht aus 5 Dateien. Drei Dateien sind die Implementierungen von Methoden der verwendeten Bibliotheken. Diese werden in Abbildung 1 dargestellt. Von der Keras Bibliothek [9] wurden die Rotations-, Shift- und Zoomfunktionen des ImageDataGenerators übernommen. Diese würden für die bessere Übersicht getrennt implementiert, können aber auch zusammen angewandt werden. Die Anzahl an zu generierenden Bildern für die Rotation wurde auf 9

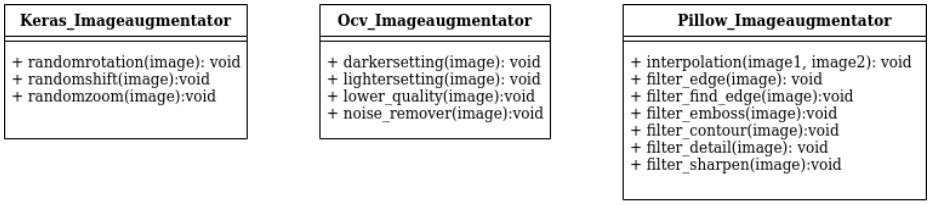

Abb. 1. Implementierte Klassen zu den verwendeten Bibliotheken

festgelegt, die für den Shift und Zoom auf 5. Mit der Zoomfunktion wird das Originalbild in verschiedene Richtungen rotiert. Ein exemplarisches Ergebnis der ausgeführten Methode findet sich in Abbildung 2 wieder. Wie zu erkennen ist, wird das Bild mal stärker und mal schwächer im Bild rotiert. Die frei geworden Flächen werden mit Farbverläufen ausgefüllt.

Abb. 2. Ergebnisse der implementierten Rotationsfunktion von Keras

Bei der Shiftfunktion wiederum wird das Bild in verschiedene Richtungen verschoben. Dadurch verschiebt sich auch das gesuchte Objekt in verschiedene Ecken des Bildes. Auch in dieser Funktion werden die frei gewordenen Flächen mit Farbverläufen ausgefüllt. Ein Beispiel für die Verschiebung findet sich in Abbildung 3 wieder.

Abb. 3. Ergebnisse der implementierten Shiftfunktion von Keras

Die Zoomfunktion generiert Bilder bei denen das Bild vergrößert oder verkleinert wird. Dadurch kann die in Abbildung 4 erkennbare Palme mal breiter und mal dünner erscheinen. Bei einem horizontalen Raus-/Reinzoomen werden die Ränder mit Farbverläufen ausgefüllt, während bei einer vertikalen nur im Falle eines Rauszoomens die Bilder mit Farbverläufen ausgefüllt werden.

Abb. 4. Ergebnisse der implementierten Zoomfunktion von Keras

Der `Ocv_Imageaugmentator` und der `Pillow_Imageaugmentator` bieten auch Möglichkeiten für die Rotation von Bildern an. Jedoch ist es weniger zeitaufwendig diese mit Keras zu rotieren, da hier auf zufällige Weise beliebig viele rotierte Duplikate vom Original-Bild erstellt werden können. `Ocv_Imageaugmentator` und der `Pillow_Imageaugmentator` werden somit verwendet um weitere Daten Augmentierungen vorzunehmen. Die Ergebnisse von `Ocv_Imageaugmentator` und `Pillow_Imageaugmentator` sind in Abbildung 5 dargestellt.

Den Klassendiagrammen zu entnehmen stammen die Bilder b-e der Klasse `Ocv_Imageaugmentator` und die Bilder f-k der Klasse `Pillow_Imageaugmentator`. Von der Pillow Bibliothek wurden vordefinierte Filter verwendet, weshalb sich diese Methoden mit geringen Aufwand implementieren ließen. Am komplexesten gestaltete sich die Interpolation. Diese erfordert zwei Bilder der selben Größe, um daraus ein neues Bild zu generieren. Da sich die Bildgröße zwischen Bildern

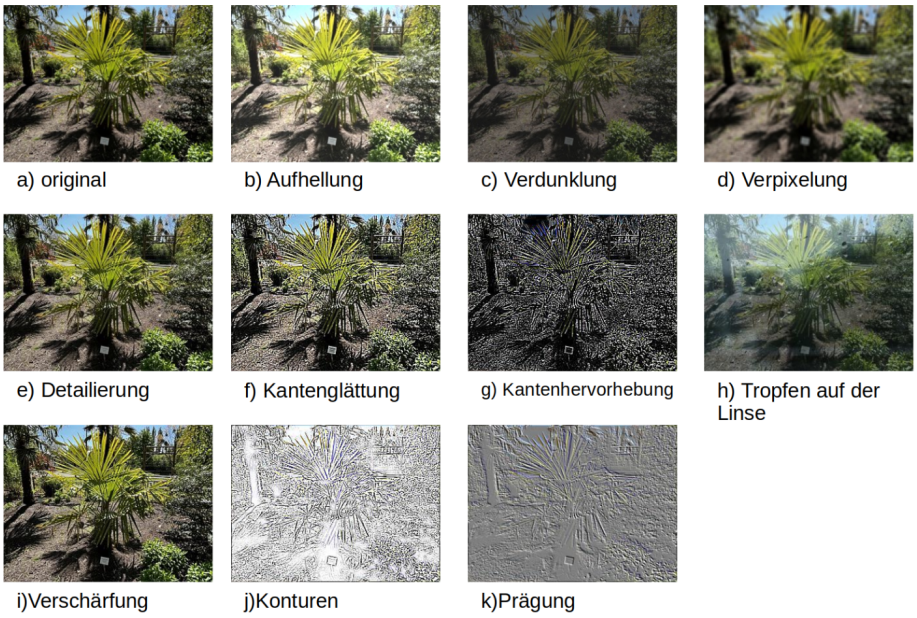

Abb. 5. Ergebnisse der implementierten Methoden von Opencv und Pillow

unterscheiden kann, wird das Bild im Quellcode dynamisch angepasst und auf die selbe Größe des an die Methode übergebenen Bildes angepasst. Auch die Suche nach einem passenden Bild ist für den Effekt wichtig. So wurde hierbei das in Abbildung 6 sichtbare Bild mit der verregneten Fensterscheibe verwendet. Das Bild wurde auf Pixabay[10] gefunden, dessen Bilder lizenzfrei zur Verfügung stehen. Dieses eignet sich hervorragend für die Interpolation, da bis auf die Regentropfen auf der Scheibe keine dominanten Objekte sichtbar sind. Die für die Interpolation bereitgestellte »blend« Funktion bietet die Möglichkeit den Fokus auf ein bestimmtes Bild zu lenken. In dieser Anwendung werden jedoch beide Bilder gleichermaßen für das neue Bild berücksichtigt. Es wird auch genau der gewünschte Effekt erzielt, sodass es so aussieht als wäre die Kamera tatsächlich durch Regen nass geworden.

Neben den drei Dateien für die Data Augmentation besteht das Programm zusätzlich noch aus einem Filter, um nach der Generierung der neuen Bilder einen Dateinamen festzulegen. Dies soll sicherstellen, dass jedes Bild noch seinem Original zuzuordnen ist.

Die Hauptklasse führt mit einer Schleife für jedes Bild in einem Quellordner die Funktionen der Klassen aus. Die Augmentatorklassen speichern die Bilder anschließend in einem Zielordner ab. Insgesamt werden durch das Programm aus einem Originalbild 30 offline augmentierte Bilder generiert.

Abb. 6. Die Interpolation um Regeneffekte auf das Bild zu bekommen. Links ist das Originalbild und mittig das von Pixabay[10] heruntergeladene Bild mit der verregneten Scheibe. Rechts ist das Ergebnis der Interpolation

6 Zusammenfassung und Ausblick

Die praktische Umsetzung des Konzepts hat gezeigt, dass sich schnell mit bereits vorhandenen Bibliotheken zusätzliche Trainingsdaten mittels offline Data Augmentation generieren lassen. Gerade die Bibliothek von Keras preprocessing eignet sich besonders dafür, schnell und ohne größeren Mehraufwand augmentierte Daten zu generieren. Hier wurden drei Methoden implementiert, welche beliebig viele augmentierte Bilder generieren könnten. Mit 30 zusätzlichen Trainigsbildern pro Original-Bild, kann so schnell ein großer Datensatz aufgebaut werden. Zusätzlich können so einige Probleme eines kleinen Datensatzes gelöst werden. Mit einer Verdunklung des Bildes kann eine schlechtere Beleuchtung, wie beispielsweise schlechteres Wetter oder Schatten, in das Training mit einfließen. Mit einer Aufhellung kann wiederum besseres Wetter mit einer stärkeren Sonneneinstrahlung nachgestellt werden. Mit Hilfe der Rotations- und Shiftfunktion ist das Bild in verschiedenen Blickwinkeln vorhanden. Auch wird durch die Zoomfunktion die Form des Objekts verändert, da es dadurch mal größer, kleiner, dünner oder breiter erscheint. Durch die Verpixelung wird berücksichtigt, dass auch Kameras mit schlechterer Qualität existieren. Die Kantenhervorhebung, Prägung und Konturen wiederum sollen den Fokus auf die Struktur der Pflanze lenken, schließlich ist gerade im Winter ausschließlich bei den meisten Pflanzen die Struktur des Geästs ersichtlich, zusätzlich wird der Hintergrund des Bildes ausgeblendet. Auch gelingt es anhand der Interpolation teilweise Regen zu simulieren.

Ein Problem, welches sich jedoch beim Training eines Objektdetektors hervorhebt, ist dass die Daten nach der Erweiterung noch mit einer Bounding-Box gelabelt werden müssen. Dies kann bei einer großen Anzahl von Bildern zu einem erheblichen Mehraufwand führen. Wären dies z.B. fünf Original-Bilder eines zu erkennenden Objekts müssten 5*30 also 150 Bilder pro Objekt manuell mit einer Bounding-Box ausgestattet werden. Es wäre folglich von Vorteil, wenn das Programm noch eine Bounding-Box selbst dem Bild hinzufügt. Dies gestaltet sich jedoch als schwierig, gerade bei der Keras Bibliothek verschiebt sich das Objekt sehr stark im Bild und die Bounding-Box müsste mit rotiert, verschoben, vergrößert oder verkleinert werden. Jedoch geschieht die Rotation zufällig, dadurch

ist es nicht möglich die Bounding-Box gleichermaßen mit zu verschieben. Auch gelingt es durch reine offline Data Augmentation nicht alle problematischen Fälle abzudecken. Beispielsweise kann der Status von Pflanzen im Winter nur bedingt berücksichtigt werden. Zwar legen die augmentierten Bilder der Kantenhervorhebung den Fokus auf das Geäst der Pflanzen, jedoch werden hier auch die Blätter, welche markant sind, nicht komplett aus dem Bild genommen. Des weiteren muss untersucht werden, ob die zusätzlichen augmentieren Daten auch wirklich zu einer Verbesserung des Trainings führen. Schließlich stammen die Bilder trotzdem noch von einem Original-Bild ab, welches somit in einer mehr oder weniger stark veränderten Form für das Training verwendet wird. So ist z.B. bei der von Keras verwendeten Bibliothek für die Data Augmentation der Hintergrund noch gleich. Auch die Farben bleiben in einem ähnlichen Spektrum. Zwar entstehen durch die anderen Klassen noch Bilder wo die Farben des Bildes verändert werden, oder z.B. bei der Kantenhervorhebung der Hintergrund keine Rolle spielt, jedoch sind immernoch viele Bilder mit dem selben Hintergrund beim Training vorhanden. Dies dürfe im Kontext der Erkennung in einem botanischen Garten ein geringeres Problem sein, als wenn die Pflanze allgemein erkannt werden soll. Allerdings kommen die selben Pflanzen auch mehrfach im botanischen Garten vor, sodass zumindest die anderen Pflanzen von anderen Standorten auch aufgenommen werden müssen. Folglich ersetzt die Data Augmentation den Prozess des Sammelns von original Bildern nicht komplett außer Kraft. Gerade die Blütenbildung von bestimmten Pflanzenarten wie Obstbäumen können nicht mit augmentierten Daten simuliert werden.

Aus den beschriebenen Problemen des vorherigen Abschnitts ergeben sich für die weitere Arbeit die in den nächsten Zeilen beschriebenen nächsten Schritte für das Projekt. Zunächst müssen auch die Jahreszeiten und verschiedenen Formen von Pflanzen noch abgedeckt werden. Hier könnte online auf Seiten wie Pixabay noch nach weiteren Bildern der zu erkennenden Objekten gesucht werden. Des Weiteren könnte ein GAN oder eine Rendering-Pipeline aufgebaut werden, um synthetische Bilder für die Objekte zu generieren, welche z.B. auch die Jahreszeiten abdecken. Der nächste Schritt stellt primär jedoch das Austesten der Auswirkung der erstellten augmentierten Daten auf das Training dar. Um den Schritt des Labelns von Daten zu vereinfachen und um nicht zu einem erheblichen Mehraufwand zu führen, sollte das in dieser Arbeit verwendete Programm noch um einen Erhalt der Bounding-Box bei den Bildern erweitert werden, welche ihre Rotation nicht verändern.

Literatur

1. Apple: Create ml (2022), `https://developer.apple.com/machine-learning/create-ml/`
2. Apple: Creating an image classifier modell (2022), `https://developer.apple.com/documentation/createml/creating-an-image-classifier-model`
3. Bhattarai, B., Baek, S., Bodur, R., Kim, T.K.: Sampling strategies for gan synthetic data. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2303–2307. IEEE (2020)

4. Borkman, S., Crespi, A., Dhakad, S., Ganguly, S., Hogins, J., Jhang, Y.C., Kamalzadeh, M., Li, B., Leal, S., Parisi, P., et al.: Unity perception: Generate synthetic data for computer vision. arXiv preprint arXiv:2107.04259 (2021)
5. Cao, T.H.: Video-based facial expression recognition with deep learning (2022), https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/cao.pdf
6. Eggert, C., Winschel, A., Lienhart, R.: On the benefit of synthetic data for company logo detection. In: Proceedings of the 23rd ACM international conference on Multimedia. pp. 1283–1286 (2015)
7. Eggert, D.: Entwicklung zweier setups für eine augmented reality anwendung zum erkennen von obst- und gemüsesorten mit neuronalen netzen (2022), https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2022-proj/eggert_gp.pdf
8. Fredrik Lundh, ..A.C., Contributors: Pillow (2022), https://pillow.readthedocs.io/en/stable/
9. Google: tf.keras.preprocessing.image.imagedatagenerator (2022), https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
10. Jusko: weather-glas-rain-wolke-cloudy-day-2400074 (2017), https://pixabay.com/de/photos/weather-glas-rain-wolke-cloudy-day-2400074/
11. Kamilaris, A., Brink, C.v.d., Karatsiolis, S.: Training deep learning models via synthetic data: application in unmanned aerial vehicles. In: International Conference on Computer Analysis of Images and Patterns. pp. 81–90. Springer (2019)
12. Kaushik, D., Hovy, E., Lipton, Z.C.: Learning the difference that makes a difference with counterfactually-augmented data. arXiv preprint arXiv:1909.12434 (2019)
13. de Melo, C.M., Torralba, A., Guibas, L., DiCarlo, J., Chellappa, R., Hodgins, J.: Next-generation deep learning based on simulators and synthetic data. Trends in cognitive sciences (2021)
14. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of big data **6**(1), 1–48 (2019)
15. Talukdar, J., Biswas, A., Gupta, S.: Data augmentation on synthetic images for transfer learning using deep cnns. In: 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN). pp. 215–219. IEEE (2018)
16. team, O.: Opencv (2022), https://opencv.org/
17. Volk, G., Müller, S., Von Bernuth, A., Hospach, D., Bringmann, O.: Towards robust cnn-based object detection through augmentation with synthetic rain variations. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). pp. 285–292. IEEE (2019)
18. Wang, A., Sun, Y., Kortylewski, A., Yuille, A.L.: Robust object detection under occlusion with context-aware compositionalnets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12645–12654 (2020)