Konzept zur Krankheiten Erkennung von Pflanzen per Smartphone-App

Darius Bonk (22213311)

Vipul Durgade (22213303)

Matial Domche (22213315)

Kiran Krishnakumar (22213304)

Projektbericht zur Abgabe in „Echtzeit Bildverarbeitung“

bei

Vitali Czymmek, Fachhochschule Westküste

vorgelegt am 30.01.2022

Inhalt

[Abbildungsverzeichnis II](#_Toc125639699)

[Tabellenverzeichnis III](#_Toc125639700)

[1. Einleitung 1](#_Toc125639701)

[2. Zielsetzung 2](#_Toc125639702)

[3. Rahmenbedingungen 2](#_Toc125639703)

[4. Beschreibung der Durchführung 3](#_Toc125639704)

[4.1 Aufbau der Bilder und des Datensatzes 3](#_Toc125639705)

[4.2 Bearbeitung des Datensatzes 4](#_Toc125639706)

[4.3 Modellentscheidung 5](#_Toc125639707)

[5. Android App 7](#_Toc125639708)

[6. Evaluation 8](#_Toc125639709)

[6.1 Testszenarien 8](#_Toc125639710)

[6.2 Tensorboard 14](#_Toc125639711)

[7. Zusammenfassung und Ausblick 18](#_Toc125639712)

[Literaturverzeichnis i](#_Toc125639713)

[Anhänge ii](#_Toc125639714)

# Abbildungsverzeichnis

[Abbildung 4.1‑1: Durchschnittliche Ernteverluste durch Insekten (grün), Krankheiten (rot) und Unkräutern (blau) von ausgewählten Pflanzen. 1](#_Toc125639750)

[Abbildung 4.2‑1: Vertikal. Abbildung 4.2‑2: Spiegelverkehrt. Abbildung 4.2‑3: Gedreht. 4](#_Toc125639751)

[Abbildung 4.2‑4: Nutzeroberfläche LabelImg mit Beispiel. 5](#_Toc125639752)

[Abbildung 4.3‑1: IoU-Verhältnis grafische Darstellung. 6](#_Toc125639753)

[Abbildung 4.3‑1: App Logo Plant Disease Recognition. Quelle: Eigene Darstellung. 7](#_Toc125639754)

[Abbildung 6.2‑1: Evaluation in Tensorboard eines gesunden Apfelblattes. 14](#_Toc125639755)

[Abbildung 6.2‑2:Evaluation in Tensorboard eines erkrankten Kartoffelblattes. 15](#_Toc125639756)

[Abbildung 6.2‑3: Evaluation in Tensorboard eines erkrankten Apfelblattes. 15](#_Toc125639757)

[Abbildung 6.2‑4: Lernrate des ausgewählten Modelles über die Zeit. 16](#_Toc125639758)

[Abbildung 6.2‑5: Total\_loss des ausgewählten Modelles über die Zeit. 16](#_Toc125639759)

[Abbildung 6.2‑6: Regularization\_loss des ausgewählten Modells. 17](#_Toc125639760)

# Tabellenverzeichnis

[Tabelle 4.3‑1: Precision Score für das 320x320 px Modell 5](#_Toc125639851)

[Tabelle 4.3‑2: Precision Score für das 640x640 px Modell 6](#_Toc125639852)

[Tabelle 4.3‑3: Precision Score für das 320x320 px Modell 6](#_Toc125639853)

[Tabelle 6.1‑1: Testszenario Apfel 9](#_Toc125639854)

[Tabelle 6.1‑2: Testszenario Kartoffel 10](#_Toc125639855)

[Tabelle 6.1‑3: Testszenario Tomate 11](#_Toc125639856)

# Einleitung

Der Verlust an Ernte in der Landwirtschaft setzt sich aus verschiedenen Faktoren zusammen (Abbildung 1-1). Zum einen aus Insekten und Unkräutern, zum anderen aus Krankheiten, die die Pflanzen selbst befallen und sich auf andere Pflanzen übertragen kann, wodurch diese ebenfalls verenden.

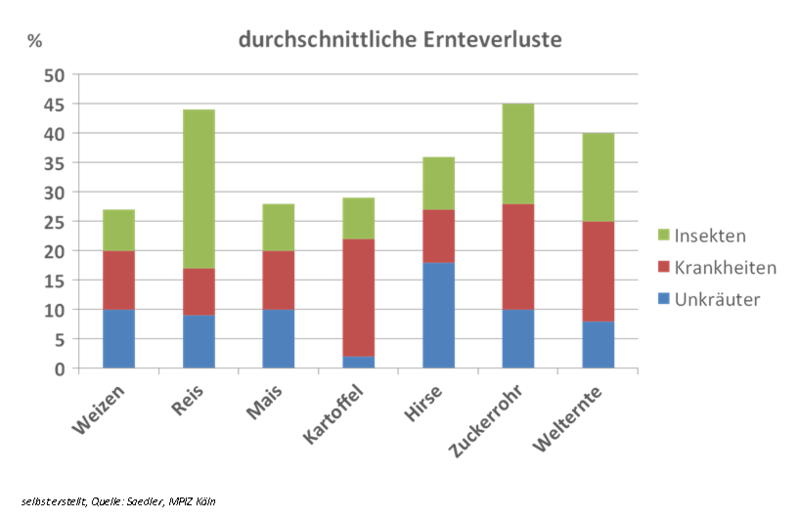


Abbildung ‑: Durchschnittliche Ernteverluste durch Insekten (grün), Krankheiten (rot) und Unkräutern (blau) von ausgewählten Pflanzen.  
Quelle: (Wikipedia, 2022)

Die Ernteverluste in den verschiedenen Kategorien können jedoch weiterhin reduziert werden, in dieser Arbeit wird eine Methode zur Reduzierung der Verluste durch frühzeitige Erkennung von Krankheitsbefall der Pflanzen entwickelt. Durch das Entwickeln einer Applikation wird die Zeit reduziert, um erkrankte Pflanzen zu erkennen Krankheitserkennung ist es möglich den Ernteverlust durch Krankheiten zu minimieren. Dadurch kann zusätzlich Geld und Zeit gespart werden, da Prüfungsprozesse verkürzt werden können.

# Zielsetzung

Die Applikation soll dem Nutzer ermöglichen ohne jegliche biologischen Vorkenntnisse, Pflanzenkrankheiten von Äpfeln, Kartoffeln, Korn und Tomaten zuverlässig und unter den gegebenen Rahmenbedingungen, zu erkennen.

Von zusätzlicher Relevanz ist dabei, dass die Applikation ohne Internet funktionieren kann, da eine Internetverbindung auf dem Agrarfeldern nicht vollständig gewährleistet sein kann.

Außerdem wird die Ausgabe der Applikation zum besseren Verständnis auf Englisch ausgeführt.

# Rahmenbedingungen

Aufgrund des Datensatzes ergeben sich einige Rahmenbedingungen, die Festgelegt sind, um die Verwendung der App zu ermöglichen. Die App wird nur für Mobiltelefone mit dem Android Betriebssystem entwickelt. Die relevanteste Bedingung ist, dass hinter den Blättern ein Blatt Papier gehalten werden muss, damit die App die kranken Blätter erkennt. Der Datensatz muss größtenteils verwendet werden, da sich zu Beginn des Labelings die Saison für die meisten Pflanzen in der Dithmarschen Region zu Ende ist bzw. Viele Pflanzen im Datensatz sind nicht in Dithmarschen vorhanden. In der Regel wurden ein Datensatz mit gesunden Blättern als Referenz und, bis auf bei dem Datensatz von Kartoffeln, wurden immer drei verschiedene Krankheiten zur Erkennung verwendet.

Da der Datensatz auf 600 Bilder von gesundem und krankem Korn limitiert ist werden die anderen Pflanzen und Krankheiten auch auf 600 Bilder limitiert, um eine ausgeglichene Bewertung zu ermöglichen.

# Beschreibung der Durchführung

## Aufbau der Bilder und des Datensatzes

Da die Bilder aus einem vorgefertigten Datensatz stammen (Bhattarai, 2022) in den mehrere verschiedenen Krankheiten vorhanden waren wurden sich unter den Gruppenmitglieder geeinigt, dass Pflanzen verwendet, werden die es in der Region Dithmarschen um die Fachhochschule Westküste gibt. Es wurde sich für Äpfel, Mais, Tomaten und Kartoffel mit den folgenden Daten entschieden:

* Gesunde Bilder aller Pflanzen
* **Apfel**: schwarze Apfelfäule, Apfelschorf, Cedar Apfel
* **Tomate**: Blattschimmel, Mosaikvirus, Bakterienpunkte
* **Mais:** Maisrost, graue Blattfleckenkrankheit, nördliche Blattfäule
* **Kartoffel:** leichte Blattfäule, frühe Blattfäule

Bei jeder Krankheit wurden 600 Bilder zum Training verwendet zusammen mit den daraus entstehenden .XML-Dateien stand dem Neuronalen Netz knapp 1200 Dateien pro Krankheit zur Verfügung. Ausgenommen davon sind die schwarze Apfelfäule sowie die nördliche Blattfäule von Mais. Bei den beiden Krankheiten traten Komplikationen mit den Dateien und dem Training auf wodurch sie für den weiteren Verlauf des Projektes geschnitten wurde.

## Bearbeitung des Datensatzes

Der Datensatz von der Website Kaggle.de beinhaltete die Bilder mit einer Bildauflösung von 256x256 px. Zudem stehen in dem Datensatz Bilder der einzelnen gewählten Krankheiten die Bilder in verschiedenen Formaten wie z.B. vertikal (Abbildung 4-1), spiegelverkehrt (Abbildung 4-2) oder gedreht (Abbildung 4-3) zur Verfügung.

Abbildung ‑: Vertikal. Abbildung ‑: Spiegelverkehrt. Abbildung ‑: Gedreht.  
Quelle: Bhattarai, 2022

Bei den Krankheitsbildern aus dem Datensatz handelt es sich nur um die einzelnen Bilder um nun die Bilder mit den dazugehörigen Labeln (A1) zu versehen wird das Programmen LabelImg verwendet.

Bei dem Programm LabelImg (Abbildung 4.2-4) wird ein Bild in das interface geladen. Mit der Funktion „createRectbox“ wird das hochgeladen Bild mit dem entsprechenden Label der mit der Label-Map (A1) korrespondiert versehen. Sodass das Neuronale Netz auch bei einer größeren Menge an verschiedenen Blättern in der Lage ist diese zu unterscheiden und die richtige Krankheit zu identifizieren. Bei den Bildern wurde, wie bereits in 4.1 erwähnt, versucht eine gleichmäßige Anzahl zu verwenden sowie möglichst viele Bilder in den verschiedenen Variationen sowie mit unterschiedlichen Lichtverhältnissen.

Da die Blätter auf einem Hintergrund liegen wird versucht diesen zu minimieren. Durch die Minimierung wird, die Leistungsfähigkeit und Zuverlässigkeit des Neuronalen Netzes verbessert.

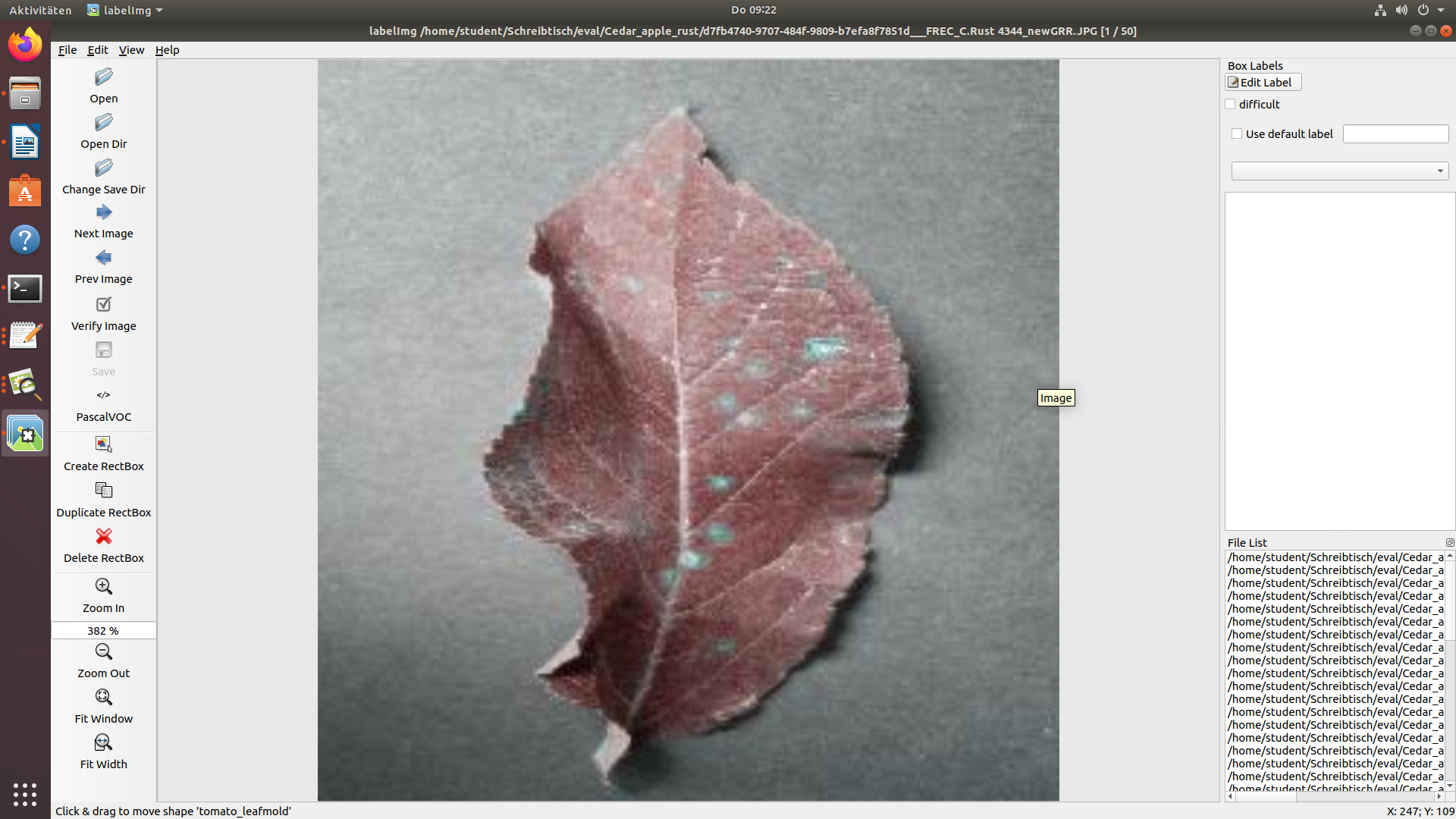


Abbildung ‑: Nutzeroberfläche LabelImg mit Beispiel.

Für das weitere Training des Neuronalen Netzes wurde Tensorflow 2.7 inkl. ObjectDetection API 0.1 verwendet (Vladimirov, 2023).

## Modellentscheidung

Um ein geeignetes Modell zu finden, wurden mehrere Versionen des „ my\_ssd\_mobilenet50\_v2\_fpnlite“-Modells verwendet. Bei dem Training wurden die Bilder auf verschiedene Bildauflösungen hochskaliert. Diese hochskalierten Bilder werden im weiteren Trainingsverlauf des Neuronalen Netzes weiterverwendet. Dies hat für diese Anwendung jedoch kaum Einfluss auf die Precision des fertigen Modells (vgl. Tabelle 2 und Tabelle 3).

Durch Veränderung der „Pipeline.config“ können die Parameter des Modelles, je nach notwendigkeit verändert werden. Eine Auswirkung auf das Ergebnis und die Precision des Modelles, wird jedoch erst nach Abschluss des Trainings ersichtlich.

Für die Modelle wurden neben den verschiedenen Auflösungen auch mit unterschiedlichen Parametern getestet. Das Mobilenet-Modell mit einer Auflösung von 320x320 px (Tabelle 2), einer Batchsize von vier, 50000 Lernschritten sowie neun zu identifizierenden Label (Anhang 1) aus der Label-Map (Anhang 2).

Tabelle ‑: Precision Score für das 320x320 px Modell

|  |  |  |  |
| --- | --- | --- | --- |
| IoU | Area | maxDets | Precision(mAp) |
| 0.50:0.95 | All | 100 | 0.703 |
| 0.50 | All | 100 | 0. |
| 0.75 | All | 100 | 0.878 |

Das 640x640 px Modell (Tabelle 3) unterscheidet sich in der Batchsize, die hier acht beträgt. Sowie in der Vergrößerung der Bilder auf 640x640 px (Anhang 3).

Tabelle ‑: Precision Score für das 640x640 px Modell

|  |  |  |  |
| --- | --- | --- | --- |
| IoU | Area | maxDets | Precision(mAp) |
| 0.50:0.95 | All | 100 | 0.771 |
| 0.50 | All | 100 | 0.884 |
| 0.75 | All | 100 | 0.860 |

Das dritte getestete Modell ist auch ein 320x320 px Modell mit einer anderen Batchsize von acht (Anhang 4).

Tabelle ‑: Precision Score für das 320x320 px Modell

|  |  |  |  |
| --- | --- | --- | --- |
| IoU | Area | maxDets | Precision(mAp) |
| 0.50:0.95 | All | 100 | 0.766 |
| 0.50 | All | 100 | 0.892 |
| 0.75 | All | 100 | 0.858 |

In den drei Tabellen wird außerdem ersichtlich, dass sie die Precision je nach Intersection-over-Union-Wert (IoU) verändert. Der IoU-Wert (Abbildung 4-5) beschreibt dabei einen Schwellenwert, ob das vorhergesagte Ergebnis ein True Positiv, also einen Fall bei dem das Modell die Klasse richtig vorhergesagt hat. Sowie einem False Positiv, also einem Fall, indem das Modell die positive Klasse nicht richtig vorhergesagt hat.

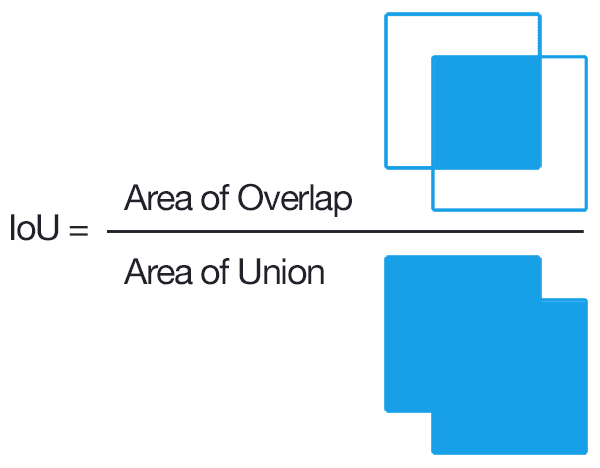


Abbildung ‑: IoU-Verhältnis grafische Darstellung.  
Quelle: https://b2633864.smushcdn.com/2633864/wp-content/uploads/2016/09/iou\_equation.png?lossy=1&strip=1&webp=1

Dabei beschreibt der Zähler den Wert indem sich die Handgezeichneten Labelbox mit der von dem Modell erstellten Box überschneiden (Area of Overlap, 1). Der Nenner beschreibt dabei die gesamte Fläche, die die handgezeichnete und die von dem Modell erstelle Box einnehmen (Area of Union, 2). Das heißt, wenn sich die beiden Boxen exakt überschneiden, ist IoU = 1 und wenn sie sich gar nicht überschneiden ist IoU = 0. In der ersten Spalte der drei Tabellen ist die Mindestwert des IoUs für die Boxen angegeben in der letzten Spalte ist die daraus resultierende Precision Score, der zur Bewertung des Modells relevant ist.

Durch den direkten Vergleich wird ersichtlich, dass das Modell mit der 320x320 px Bildauflösung den besten Precision Score aller getesteten Modelle erreicht. Zusätzlich führt eine Vergrößerung der Bildauflösung dazu, dass sich die Precision verringert. Eine vergrößerung der Batchsize führt zu einer zusätzlichen verringerung des precision Scores, da somit eine größere Anzahl an Proben pro Trainingsdurchlauf verwendet werden. Dadurch wurde das 320x320 px Modell ausgewählt, um die App zu realisieren.

# Android App

Logo

Description automatically generated with low confidence

Abbildung ‑: App Logo Plant Disease Recognition.  
Quelle: Eigene Darstellung.

Für die Programmierung der App wird die „TensorFlow Lite Object Detection Android Demo“- Applikation verwendet. Beim Generieren der App kann nicht der endgültige Checkpoint des Modelles verwendet werden, sondern ein vorläufiger Checkpoint wird als Protokollbufferdatei, zu einem Zwischenmodell, welches mit dem TFlite-Konverter kompatible ist exportiert. Für die Umwandlung eines TFLite-Modells wird die Python API verwendet.

Beim Exportieren eines Modelles ist es möglich das Modell zu quantisieren, wodurch einerseits die Dateigröße verringert wird und andererseits die Latenz verringert werden kann. Da das vorhandene Modell eine vernünftige Größe hat wird kein Post-Training-Quantisierung durchgeführt. Um das exportierte Modell nun in der App zu verwenden werden die Label und die Metadaten des trainierten Modells mit dem Metadaten-Writer zu TFLite hinzugefügt.

Um Änderungen an der App vorzunehmen, wird Android Studio verwendet. Die vorhandenen Modelle in der TFLite app werden entfernt und das Modell zur Krankheitserkennung der Pflanzen wird in dem Verzeichnis „Assets“ hinzugefügt. Für die Modellsimulation in den Nachfolgenden Kapiteln wird mit dem „Redmi Note 10“ Handy durchgeführt.

# Evaluation

## Testszenarien

Um die Genauigkeit der App in der realen Welt zu testen, wurden verschiedene Testszenarien entwickelt. Es wurden dabei dem Modell unbekannte Bilder aus dem vorhanden Datensatz oder Internet verwendet. Der Grund für die Verwendung von Bildern aus dem Datensatz oder Internet ist die Tatsache, dass das Projekt im Herbst/Winter gestartet wurde. In diesem Zeitraum führen die meisten der Untersuchten Pflanzen keine Blätter mehr.

Zudem wurde untersucht, wie sich die App bei Unbekannten Formen und Gegenständen verhält. Werden diese nicht erkannt oder als Blätter identifiziert.

Tabelle ‑: Testszenario Apfel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apfel | | | | |
| Szenario | Genauigkeit | Auffälligkeiten | Bild | Erfolgreich |
| Apfel Healthy | 66 – 72% | - | A screenshot of a computer  Description automatically generated with medium confidence A picture containing text, plant  Description automatically generated | Checkmark with solid fill |
| Apfel Cedar | 90 - 94% | - | A screenshot of a computer  Description automatically generated with low confidenceA picture containing text  Description automatically generated | Checkmark with solid fill |
| Apfel Blackrot | 88 - 94% | - | A picture containing map  Description automatically generatedA screenshot of a cell phone  Description automatically generated with low confidence | Checkmark with solid fill |

Tabelle ‑: Testszenario Kartoffel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Potato | | | | |
| Szenario | Genauigkeit | Auffälligkeiten | Bild | Erfolgreich |
| Potato Healthy | 77 – 87 % |  | A picture containing text  Description automatically generatedA picture containing text, vegetable  Description automatically generated | Checkmark with solid fill |
| Potato Earlyblight | 67 – 81% |  | A picture containing map  Description automatically generatedMap  Description automatically generated | Checkmark with solid fill |
| Potato Lateblight | 77 – 88% |  | A picture containing text, leaf  Description automatically generatedA picture containing map  Description automatically generated | Checkmark with solid fill |

Tabelle ‑: Testszenario Tomate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tomato | | | | |
| Szenario | Genauigkeit | Auffälligkeiten | Bild | Erfolgreich |
| Tomato Healthy | 75 – 92% |  | A picture containing text, vegetable  Description automatically generatedA screenshot of a video game  Description automatically generated with medium confidence | Checkmark with solid fill |
| Tomato Bacterialspot | 81 – 85% |  | A screenshot of a video game  Description automatically generated with low confidenceMap  Description automatically generated | Checkmark with solid fill |
| Tomato Leafmold | 84 – 87% |  | A picture containing text, insect  Description automatically generatedEin Bild, das Text, drinnen, Screenshot, Anzeige enthält.  Automatisch generierte Beschreibung | Checkmark with solid fill |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fehlerkennungen** | | | | |
| **Szenario** | Genauigkeit | Auffälligkeiten | Bild | Erfolgreich |
| **Foreign Object-apple** | 83% | Erkennung eines Apfels als Blatt mit Krankheit. |  |  |
| **Apple leaf** | 81% | Erkennung eines gesunden Apfelblattes als ein mit Black rot erkrankten Blattes. |  |  |
| **Multiple leafs** | 61% | Erkennung nur als gesamtes, nicht eines einzelnen Blattes. |  |  |

Insgesamt zeichnet sich durch die Testszenarien ein funktionierendes Modell ab. Die Erkennung der Blätter mit Krankheiten ist in den Testszenarien mit mindesten 65 % im akzeptablen Bereich. Mithilfe der Testszenarien zeigt sich, dass das Modell in der Lage ist einige Blätter mit ihren Krankheiten auch unter einem neuen Hintergrund zu identifizieren.

Die niedrige Genauigkeit der einzelnen Szenarien lässt sich auf den verminderten Datensatz von 600 Bildern zurückführen eine Vergrößerung des verwendeten Datensatzes würde die Trainingsdauer zwar verlängern aber würde zu einer Erhöhung der Genauigkeit führen. Zudem ist es der Erkennung innerhalb der App nicht möglich einzelne Blätter zu identifizieren. Dies ist darauf zurückzuführen, dass der Datensatz nur aus einzelnen Blättern besteht und die Label der Bilder immer nur ein einzelnes Blatt betrachten.

## Tensorboard

Zur Evaluation des Modelles wird das „Tensorboard“ verwendet. Als Trainingsdaten wird der letzte Checkpoint des Modelles hochgeladen und als Evaluation Daten werden die vorher getrennten 10 % der Bilder verwendet. Tensorboard ermöglicht mithilfe der Evaluation Daten eine Analyse und eine Optimierung des Modelles. Die Ausgaben in Tensorboards bestehen aus den verschiedenen Graphen zur Gesamtenverlustfunktion, dem Regularisierungsfehler, der Lernrate, dem Lokalisationsfehler sowie dem Klassifikationsfehler. Die Analyse und Auswertung der Graphen erlaubt eine Optimierung der „Pipeline.config“ des verwendeten Modelles.



Abbildung ‑: Evaluation in Tensorboard eines gesunden Apfelblattes.

Innerhalb Tensorboards ist es möglich die vorhergesagten Bilder mit Detection Box und den Labeln mit Detection Box anzuzeigen. Die Abbildung 6.2-1 zeigt ein erstes Beispiel anhand eines gesunden Apfelblattes.

Die Modellvorhersage, im linken Teil des Bildes, zeigt ein Ergebnis von 85 % das dies ein gesundes Blatt ist. Das Label im rechtem teil des Bildes zeigt das dies händisch zu 100 % als gesundes Blatt identifiziert wurde.

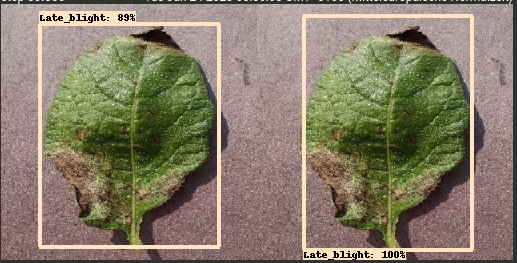


Abbildung ‑:Evaluation in Tensorboard eines erkrankten Kartoffelblattes.

Um eventuelle Fehler in der Erkennung zu identifizieren, werden auch die erkrankten Pflanzen untersucht. Abbildung 6.2-2 zeigt die richtige Identifikation eines Kartoffelblattes, welches an einer späten Fäule erkrankt ist. Es zeigt sich eine Precision von 89 % in der Erkennung des Blattes.

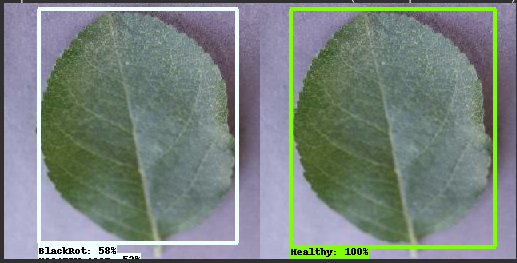


Abbildung ‑: Evaluation in Tensorboard eines erkrankten Apfelblattes.

Im Gegensatz zu den Abbildungen 6.2-1 und 6.2-2 zeigt die Abbildung 6.2-3, dass die Vorhersage eines gesunden Blattes nicht immer Fehlerfrei ist. Im rechten Teil der Abbildung zeigt sich, dass die Modellvorhersage das Blatt als ein mit schwarzer Fäule erkranktes Blatt erkennt. Das Label aus dem rechten Teil bestätigt diese Vorhersage nicht. Das Blatt stammte aus dem gesunden Datensatz und wurde als ein solches gelabelt.

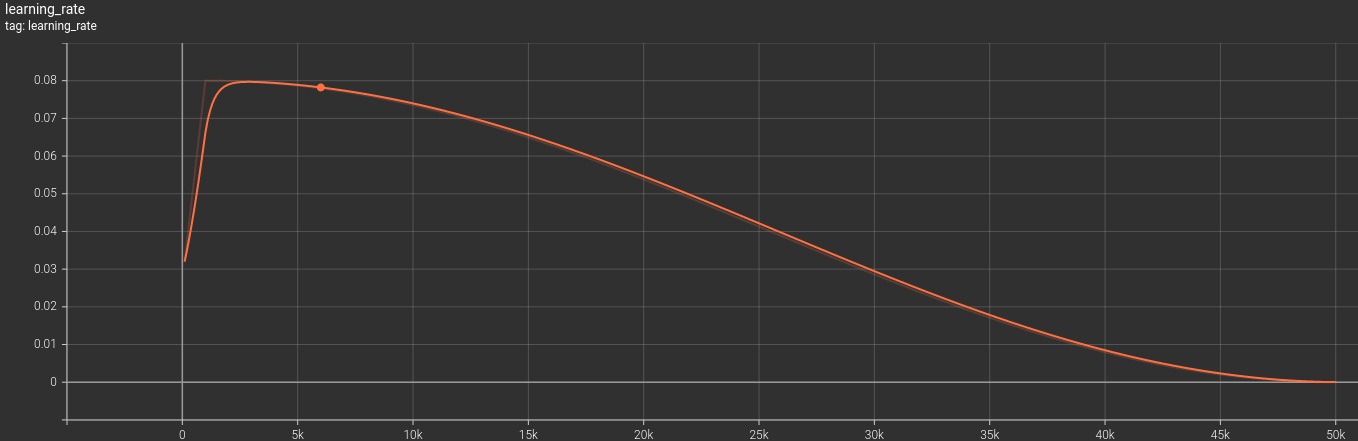


Abbildung ‑: Lernrate des ausgewählten Modelles über die Zeit.

Abbildung 6.2-4 zeigt die Variation der Lernrate des 320x320 px Modelles über die 50000 Schritte hinweg. Die Lernrate sagt aus, wie stark das Neuronale Netz die Gewichtung der einzelnen Neuronen mit Bezug auf die erkannten Fehler anpasst. Aus der Abbildung wird ersichtlich, dass in den ersten 5000 Schritten die Lernrate ihr Maximum erreicht und danach abnimmt, bis sie am Ende des Trainings gleich 0 ist.

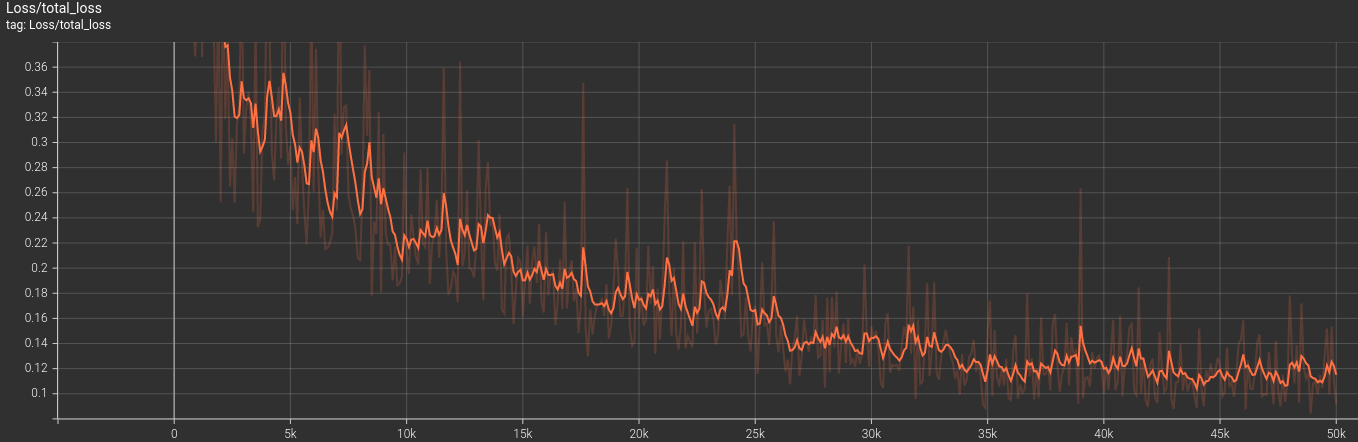


Abbildung ‑: Total\_loss des ausgewählten Modelles über die Zeit.

Bei dem Training eines Neuronalen Netzes ist es ein geringer Datenverlust zu präferier damit die Vorhersagen möglichst fehlerfrei getätigt werden. Mit der Minimierung des in Abbildung 6.2-5 dargestellten Gesamtverlustes (Total loss) verbessert das Modell seine Fähigkeiten, dass die Modellvorhersagen mit den Trainingsdaten übereinstimmen. Eine hohe Übereinstimmung ist zwar erwünscht führt aber zum Overfitting.

Bei Overfitting ist das Neuronale Netz zu gut an and die Trainingsdaten angepasst und hat Schwierigkeiten in der Abstraktion. Somit kann das ermitteln von neuen Daten ungenau und somit unbrauchbar sein.

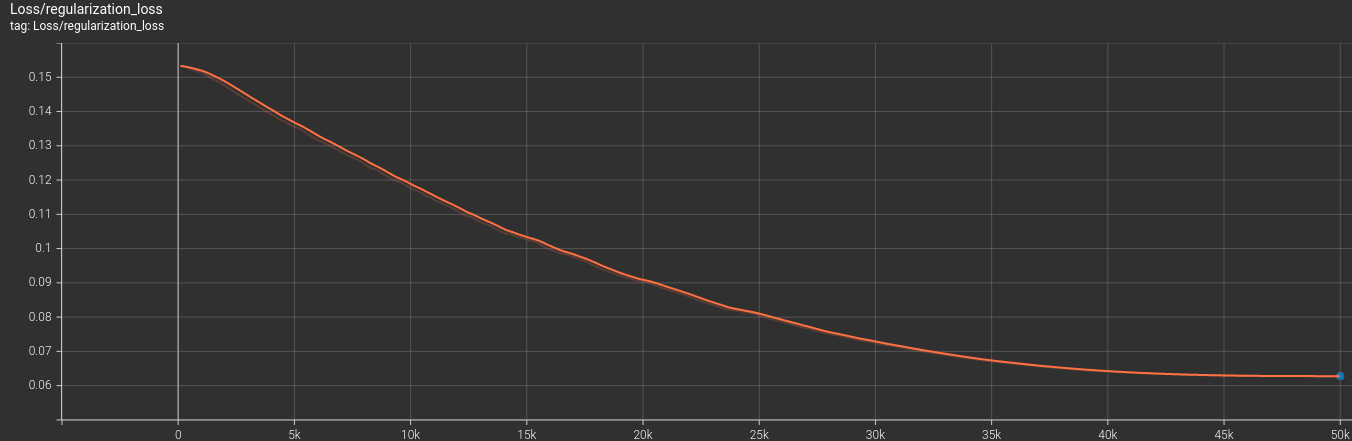


Abbildung ‑: Regularization\_loss des ausgewählten Modells.

Zur Optimierung der Generalisierung des Modelles wird der Loss-Funktion ein Regularisierung-Teil hinzugefügt. Dieser Regular Regularisierung-Teil reduziert den Gesamtfehler eines Modells und versucht außerdem das Overfitting zu minimieren bzw. zu verhindern. Abbildung 6.2-6 zeigt die Regularisierungsfehler der dem Modell hinzugefügt wird, um die Generalisierung des Modelles zu verbessern.

# Zusammenfassung und Ausblick

Bei einem ersten Trainingsdurchlauf traten unerwartete Komplikationen mit der Dauer des Trainings auf. Dieses erste Training wurde mit allen Krankheiten, labeln, und den Standardeinstellungen des Modells durchgeführt. Jedoch ergab das Training auch nach mehreren Wochen kein Ergebnis und überlastetet die CPU auf über 100 %. Durch dieses Ereignis wurde beschlossen den Mais und jeweils eine Krankheit pro Pflanze aus dem Training zu entfernen. Die weiteren Trainings verliefen ohne weitere Komplikationen.

Als weitere Begrenzung beim Verwenden stellte sich heraus, dass aufgrund der kleinen Bildauflösung des Datensatzes das Mobiltelefon sehr nah an den eigentlichen Blatt gehalten muss bzw. Bilder vergrößert werden müssen, um eine Erkennung zu ermöglichen. Diese Begrenzung wurde jedoch als akzeptabel identifiziert da die Durchführung auf weite Entfernung als unrealistisch erscheint.

Die Tatsache das es mit der App nicht möglich ist einzelne Blätter aus einem Strauch zu erkennen ist für die Verwendung nicht relevant. Es würde eine korrekte Identifikation der Krankheit der Blätter genügen. Zusätzlich ist es sowieso vonnöten das Mobiltelefon möglichst nach an die Blätter zuführen, wodurch in der Regel nur ein Blatt identifiziert werden sollte.

Zusammenfassend ist es möglich mehrere Krankheiten gemeinsam mit einem Modell zu identifizieren die Schwierigkeit liegt hierbei in einer gleichmäßigen Gewichtung der einzelnen Bilder da ein Ungleichgewicht dazu führen kann, dass eine Krankheit weitaus besser und häufiger identifiziert werden kann. Zudem beschränkt die Tatsache, dass der Datensatz mit einem neutralen Hintergrund vorbelastet ist, die gesamte Precision des Modells erheblich, wenn krankheitsbefallene Blätter ohne diesen neutralen Hintergrund untersucht werden.

In Hinblick auf die Weiterentwicklung der App kann die Funktion zur Erkennung der Krankheiten als Teil der intelligenten Agrarwirtschaft verwendet werden . Um dies zu realisieren können autonome Agrarroboter mit der Erkennung ausgestattet werden und den Anwender die Daten sowie mögliche Gegenmaßnahmen zusenden. Zusätzlich sollten mehrere Krankheiten für verschiedene Pflanzen hinzugefügt werden.

# Literaturverzeichnis

Bhattarai, S. (01. 11 2022). *Kaggle.com*. Von https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset abgerufen

Frochte, J. (2019). *Maschinelles Lernen.* München: Carl Hanser Verlag.

Github. (07. 05 2021). *Github*. Von https://github.com/tensorflow/models/blob/master/research/object\_detection/g3doc/tf2\_detection\_zoo.md abgerufen

Vladimirov, L. (24. 01 2023). *TensorFlow 2 ObjectDetection API*. Von https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html abgerufen

Wikipedia. (01. 11 2022). *Wikipedia*. Von https://de.m.wikipedia.org/wiki/Datei:Durchschnittliche\_Ernteverluste.PNG abgerufen

# Anhänge

Anhang 2: Label-Map

|  |  |
| --- | --- |
| **Blatt** | **label-Name** |
| Apple healthy | Apple\_Healthy |
| Apple black rot | Apple\_Blackrot |
| Apple cedar apple | Apple\_cedarApple |
| Tomato healthy | Tomato\_Healthy |
| Tomato LeafMold | Tomato\_LeafMold |
| Tomato BacterialSpot | Tomato\_BacterialSpot |
| Potato healthy | Healthy\_leaf |
| Potato LightBlight | Late\_blight |
| Potato EarlyBlight | Early\_blight |