

Praktikumsbericht
Zeitraum: 9. Januar - 30. April 2023

Werkstudent als Machine Learning Engineer bei der Duckeneers GmbH

Maksym Sevkovych
Matrikelnummer: 3330007

Prüfer & Betreuer: Prof. Dr. Ingo Steinwart

Institut für Stochastik und Anwendungen
Fachbereich Mathematik

vorgelegt an der Universität Stuttgart am 23. Juni 2023

Inhaltsverzeichnis

1 Künstliche Intelligenz bei der Duckeneers GmbH	4
1.1 Kartenerkennung	6
1.2 Texterkennung	6
2 Aufgaben während der Werkstudententätigkeit	7
2.1 Kartenerkennung	7
2.2 Arbeitsumgebung und Workflow	9
3 Persönliches Fazit	11

Einführung

Logistik ist ein grundlegender Bestandteil der heutigen Welt. Von Nahrungsmittel über Rohstoffe zu Konsumgütern; weltweit werden täglich unzählige Tonnen diverser Gegenstände transportiert. Manchmal allerdings nicht an das Ziel, an das die Lieferung eigentlich gehen sollte - und dies ist teuer. Die Duckeneers GmbH revolutioniert die Logistikbranche durch innovative, auf künstlicher Intelligenz basierende Technologien, um Fehler von Logistikunternehmen und die damit entstehenden Kosten zu minimieren.

In meiner Werkstudententätigkeit bei der Duckeneers GmbH bin ich unter Anderem zuständig für den technischen Teil, die künstliche Intelligenz. Da wir ein vergleichsweise kleines Team sind, ermöglichte es mir in der kurzen Zeit viele unterschiedliche Einblicke in die Welt eines Machine Learning Engineers zu erlangen. Von der Auswahl der Modelle, über deren Training und Auswertung bis zur Optimierung durfte ich alles miterleben.

Künstliche Intelligenz ist ein extrem weitgefassster Begriff, wohinter sich alles mögliche verstecken kann. Die Duckeneers GmbH beschränkt sich auf einen bestimmten Bereich der KI, nämlich der neuronalen Netze. Genauer gesagt werden neuronale Netze verwendet, um Informationen aus Bilddaten zu gewinnen; hierbei spricht man oft von Deep Learning. Dieser spezifische Bereich der künstlichen Intelligenz wird auch als Computer Vision bezeichnet.

1 Künstliche Intelligenz bei der Duckeneers GmbH

Wie schon kurz beschrieben, nutzt die Duckeneers GmbH künstliche Intelligenzen in Form von neuronalen Netzen, um Informationen aus Bilddaten zu gewinnen. In diesem Kapitel wollen wir uns dies etwas genauer anschauen.

Beginnen wir mit ein paar Worten zu der Idee, die zur Entstehung der Firma führte: Ein Logistikunternehmen, welches Mitgründer der Duckeneers GmbH ist, stellte fest, dass in einem ihrer Standorte eine überdurchschnittlich hohe Rate an falschen Lieferungen vorkam. Dies bedeutet, dass Lieferungen in falsche LKWs beladen und somit an das falsche Ziel gefahren wurden. Dies ist sehr kostspielig für Logistikunternehmen im Allgemeinen, da man einerseits Strafen zahlt, falls Lieferungen nicht pünktlich ankommen, und andererseits die falsche Lieferung an das eigentliche Ziel zu bringen extra Kosten aufwirft, die vom Kunden nicht gedeckt werden. So entstand der Ansatz diese Lieferungen direkt in der Lagerhalle, während der Beladung der LKWs zu überwachen. Das Ziel war offensichtlich, die Anzahl der fehlerhaften Lieferungen zu minimieren - und das automatisiert. Die Aufgabe bestand also darin, auf irgendeine Art und Weise nachzuverfolgen, ob Paletten richtig oder falsch beladen wurden.

Um diese Aufgabe zu bewältigen, botete es sich an eine künstliche Intelligenz zu konstruieren, die eben diesen Vorgang überwacht. Hierfür gab es in den letzten Jahren einige Durchbrüche im Bereich der neuronalen Netze, weshalb die Firma sich dazu entschied auf diese Karte zu setzen. Nun besteht die künstliche Intelligenz der Duckeneers GmbH aus zwei entkoppelten neuronalen transformierer Netzen (transformer neural networks), welche in zwei Stufen Informationen aus den Bilddaten herauszieht. Im ersten Schritt wird ein Objekterkennungsmodell benutzt, um Etikette auf der Palette zu erkennen, ein Beispiel ist in Abbildung 1.1 zu sehen. Im zweiten Schritt wird ein Modell verwendet, welches Text aus Bildern herauslesen kann, ein Beispiel ist in Abbildung 1.2 zu sehen. Durch diese zwei Modelle ist es dem Logistikunternehmen nun möglich, mit einem Bild der Palette Informationen über die Lieferungen digital vorliegen zu haben. Bei der Beladung der LKWs werden diese Informationen abgeglichen und falls ein Paket falsch beladen wird, so wird eine Warnung aufgeworfen.



Abbildung 1.1: Ein geschwärztes Beispiel aus dem Datensatz des Modells der Kartenerkennung. Bereitgestellt von der Duckeneers GmbH.

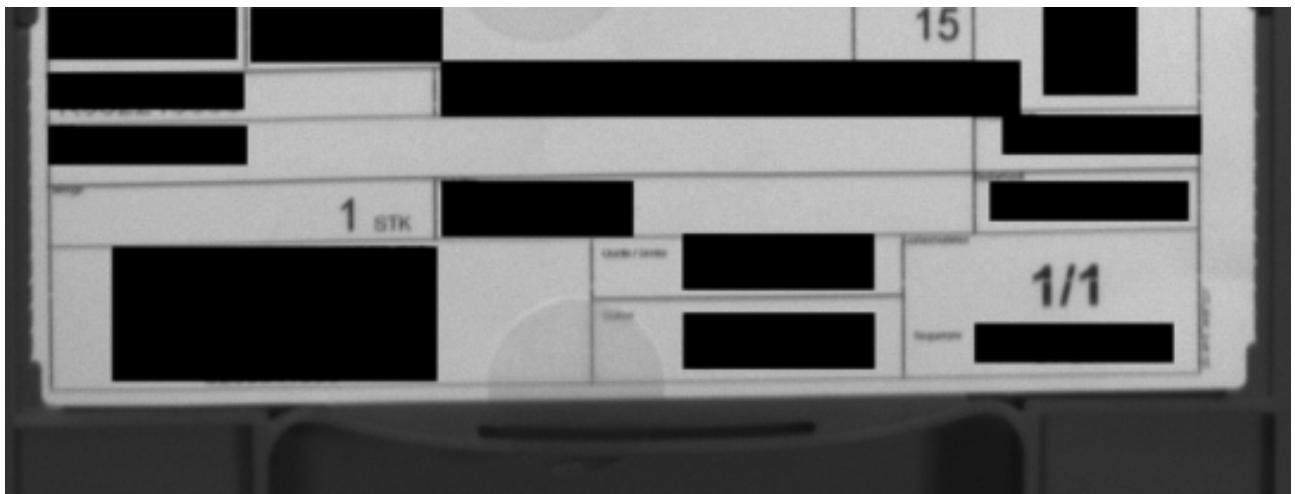


Abbildung 1.2: Ein geschwärztes Beispiel aus dem Datensatz des Modells zur Übersetzung von Bild zu Text. Bereitgestellt von der Duckeneers GmbH

Im Folgenden wollen wir uns die beiden Modelle etwas detaillierter anschauen.

1.1 Kartenerkennung

Das Modell, welches zur Erkennung der Etikette an den Paketen verwendet wird, wurde ursprünglich zur Objekterkennung entwickelt. Um genau zu sein, gab es im Laufe meiner Werkstudententätigkeit mehrere Modelle, die zu dieser Aufgabe verwendet wurden. Die Machine Learning Branche ist eine sich rasant entwickelnde, die heutigen State of the Art Technologien sind in wenigen Monaten überholt und somit obsolet. Das erste Modell, das die Duckeneers GmbH verwendete hieß Optical Character Recognition (OCR) Model. Da dies aber zu ungenau war wurden einige neue Modelle ausprobiert, bis das OCR Modell von META's Detection Transformer (DETR) Model abgelöst wurde. Dieses hielt sich einige Monate, bis auch dieses von grounding DINO und schließlich von ultralytics' YOLOv8 abgelöst wurde, welches die Duckeneers GmbH aktuell nutzt.

Bei den ganzen Bezeichnungen der eben genannten Modelle handelt es sich um sogenannte vortrainierte Modelle. Dies sind neuronale Netze, Transformer um genau zu sein, welche von den Entwicklern trainiert wurden um möglichst allgemein gut zu performen. Da viele Nutzer, wie die Duckeneers GmbH, aber nicht an allgemeiner Performance, sondern an ihrer konkreten Problemstellung interessiert sind, werden die vortrainierten Modelle nochmals nachtrainiert. Dieses Training wird dabei auf einem Datensatz durchgeführt, welcher die Problemstellung möglichst gut repräsentiert. Dies bedeutet, dass nach einem feineren Training die vortrainierten Modelle in der Lage sind präziser das Problem zu lösen und somit die Fehler des Modells minimiert. Dies ist eine Herangehensweise, die in kleinen und mittelständischen Unternehmen häufig anzutreffen ist, denn das Training solcher Modelle ist in den meisten Fällen sehr aufwändig - und damit sehr teuer. Beispielsweise wurde META's Segment Anything Model (SAM) nach eigenen Angaben drei bis fünf Tage lang mit 256 Grafikkarten mit dem A100 Chip trainiert. Würde man diese privat mieten, beispielsweise bei LambdaLabs (einem der günstigsten Cloud Computing Anbieter) für 1.20\$/Stunde pro GPU, so würde das auf ungefähr 30.000\$ hinauslaufen - für einen einzigen Trainingslauf. Man stellt schnell fest, dass dies für Unternehmen nicht realisierbar ist, sofern sie keine Giganten sind wie OpenAI, Google und Co..

1.2 Texterkennung

Die Texterkennung war mit deutlich weniger Aufwand verbunden als die Kartenerkennung. Dies hat vor Allem den Hintergrund, dass wenn das Modell zur Kartenerkennung saubere Ergebnisse liefert, das Texterkennungsmodell konsistente Daten zur Bearbeitung liefert bekommt. Die Duckeneers GmbH benutzt zur Texterkennung ein Modell namens Donut, einen OCR-free Document Understanding Transfomer - ebenfalls eine bestimmte Architektur eines neuronalen Netzes. Dieses Modell ist ebenfalls ein vortrainiertes Modell und von der Duckeneers GmbH auf deren Datensatz und damit deren Problemstellung abgestimmt. Nach einigen feinen Anpassungen hat dieses Modell eine überragende Genauigkeit von über 99%.

2 Aufgaben während der Werkstudententätigkeit

In diesem Kapitel wollen wir über die Aufgaben sprechen, welche ich während meiner Werkstudententätigkeit absolviert habe. Diese lassen sich auf mehrere größere Aufgaben aufteilen, unter Anderem Implementierung und Evaluation der Kartenerkennungsmodelle, Optimierung und Bereinigung der Trainings- und Testdaten, Optimierung der digitalen Arbeitsumgebung der Entwickler und zuletzt Einbindung und Optimierung einer GitLab CI/CD Pipeline.

2.1 Kartenerkennung

Die Kartenerkennung ist der erste und aufwändiger von beiden Schritten im Projekt. Zunächst muss ein konkretes Modell gewählt werden, dann muss es trainiert werden auf einem Datensatz, den man zuvor erstellen, bereinigen und optimieren muss. Ist das Modell fertig, so wird es in Docker Container verpackt, sodass man es nur noch aufzurufen braucht. All dies sind Aufgaben, die aus diversen Gründen ziemlich aufwändig sind. Bei einigen dieser Aufgaben durfte ich helfen, manche durfte ich sogar ganz übernehmen.

Beispielsweise befasste ich mich zunächst eine lange Zeit mit der Analyse des Datensatzes, auf dem die Modelle zur Kartenerkennung trainiert wurden. Ich erstellte Grafiken, wie die Karten im gesamten Datensatz verteilt sind und überlegte mir eine sinnvolle Aufteilung der Daten in drei Splits. Diese drei Splits unterteilen sich auf Trainingsdaten, auf welchen die Modelle trainiert werden, Testdaten, auf denen die trainierten Modelle nach jeder Epoche auf ihre Performance getestet werden, und Validierungsdaten, auf denen das fertig-trainierte Modell getestet wird. Die letzteren Daten sieht das Modell im gesamten Trainingsverlauf nicht, somit sagen sie am meisten aus, wenn über die Genauigkeit der Modelle gesprochen wird. Diese Aufteilung wird in verschiedenen Anwendungsfällen unterschiedlich generiert; in machen Fällen statisch, in manchen dynamisch. Die Duckeneers GmbH hat sich dafür entschieden diese Aufteilung statisch zu gestalten, da man dadurch mehr Kontrolle über die jeweiligen Trainingsdaten hat. Dies hat den Hintergrund, dass ungefähr 2.000 Beispielbilder, die uns zur Verfügung stehen, ein vergleichsweise kleiner Datensatz ist.

Als ich mit der ersten Analyse des Datensatzes fertig war, erstellte ich eine Pipeline mit dem aktuellen Modell, welche einen Datensatz erlangt und jedes Sample durch das Modell laufen lässt. Letztendlich konnte man dadurch automatisiert neue Modelle mit wenigen Minuten Aufwand in diese Pipeline einbinden und dadurch direkt die Ergebnisse der unterschiedlichen Modelle vergleichen. Als diese Pipeline fertig war, stellte sich die Frage wie man die Vorhersagen nun sinnvoll analysiert. Schließlich gibt es kein klares richtig oder falsch bei der Objekterkennung, da in dieser Disziplin Rechtecke, sogenannte bounding boxes, um die erkannten Objekte gezogen werden. Um also analysieren zu können, ob eine Vorhersage des Modells gut oder schlecht ist, muss man die tatsächliche bounding box mit der Vorhersage vergleichen. Hierfür wird eine

sogenannte **Intersection over Union**-Metrik eingeführt. Diese sieht aus wie folgt:

$$\mathcal{I}(A_G, A_P) = \frac{A_G \cap A_P}{A_G \cup A_P},$$

wobei $\mathcal{I} : \mathbb{N}^2 \times \mathbb{N}^2 \rightarrow [0, 1]$ die IoU-Metrik und $A_G, A_P \in \mathbb{N}^2$ die tatsächliche bounding box (ground truth), respektive vorhergesagte bounding box (prediction) beschreibt. Man stellt fest, dass die Boxen ähnlicher sind, je näher der Wert der Metrik an der 1 liegt und unterschiedlicher, je näher der Wert der Metrik an der 0 liegt. Ist der Wert 1, so sind die Boxen identisch. Ist der Wert 0, so haben sie keine Überschneidung. Schließlich muss man einen Schwellwert festlegen, ab dem eine vorhergesagte Box „ähnlich genug“ zur tatsächlichen Box ist. Diesen Schwellwert haben wir durch verschiedene empirische Herangehensweisen auf 0.8 festgelegt.

Nachdem die Metrik implementiert war, konnte man die Vorhersagen der Kartenerkennungs-Modelle beurteilen. Wir stellten eine Genauigkeit von ungefähr 94% fest - zu dem Zeitpunkt mit einer Abwandlung des DETR-Modells. Da wir mit dieser Genauigkeit nicht zufrieden waren, setzten wir uns an die Optimierung. Ich implementierte eine Pipeline zur Auswertung der Vorhersagen, die durch minimalen Aufwand die Vorhersagen der Modelle prüfte und damit die Genauigkeit der Modelle übersichtlich und anschaulich darstellte. Dadurch konnten wir direkt die Auswirkung von Änderungen sehen, egal ob am Modell oder am Datensatz.

Die nächste Aufgabe war den Datensatz erneut zu optimieren, einerseits die Qualität der einzelnen Samples und andererseits die Verteilung der Samples im Datensatz. Mit der Qualität der einzelnen Samples ist hierbei die Beschaffenheit der ground truths gemeint. Da diese Boxen von Hand gezogen wurden, sind sie mit menschlichen Fehlern behaftet. Einige Pixel Spielraum mögen zwar für den Menschen kaum einen Unterschied machen, doch sobald die Modelle auf diesem Datensatz trainiert werden, macht es letztendlich doch einen Unterschied - vor Allem da wir an Optimalität interessiert sind! Um diese kleinen Fehler zu beheben, implementierten wir eine Methode um unsere Boxen zu bereinigen. Hierfür eignete sich das grounding DINO Modell sehr gut, da die Vorhersagen so präzise waren, dass sie von menschlicher Hand nur mit Mühe reproduzierbar sind - was bei ungefähr 8.500 Boxen im Datensatz mit enormem zeitlichem Aufwand verbunden wäre. Wir ließen also das Modell Vorhersagen treffen und übernahmen die Ergebnisse als unsere neue ground truth, falls die Vorhersage den obigen Schwellwert überschritt. Damit war die Optimierung der Qualität des Datensatzes beendet, ein Beispiel der Vorhersagen des grounding DINO Modells ist zu sehen in Abbildung 2.1.



Abbildung 2.1: Ein geschwärztes Beispiel der Vorhersage des grounding DINO Modells, welche im Anschluss als neue ground truth gewählt wurde. Bereitgestellt von der Duckeneers GmbH.

Es fehlte noch die erneute Optimierung der Aufteilung. Wir unterteilten die Daten diesmal in ungefähr 200 Testdaten, 800 Trainingsdaten und 1.000 Validierungsdaten. Hierbei evaluierte ich zusätzlich die Qualität der Bilder mit unterschiedlichen Methoden, unter Anderem mit dem Python Framework cleanvision, wodurch man verschiedene Fehler wie beispielsweise Duplikate, unscharfe Bilder oder zu helle/dunkle Bilder aussortieren kann, und verschob diese in einen vierten Split, den wir Bad-Split tauften.

Als ich diese Aufgaben absolvierte, entschieden wir uns ein neues Modell auszuprobieren - das YOLOv8 Modell von ultralytics. Mit diesem Modell konnten wir nach Anpassung auf unsere Trainingsdaten eine erstaunliche Genauigkeit von ungefähr 98% erzielen, womit wir uns fürs Erste zufrieden geben.

2.2 Arbeitsumgebung und Workflow

Neben den eigentlichen Aufgaben eines Machine Learning Engineers habe ich auch einige Aufgaben bewältigt, die nicht direkt mit der Entwicklung zu tun haben. Da die Duckeneers GmbH ein recht überschaubares Startup sind und es demnach vergleichsweise wenige Mitarbeiter gibt, fallen manche Aufgaben auch in die Hände der Softwareentwickler. Da diese Aufgaben aber eher technischer Natur waren, möchte ich an dieser Stelle nicht allzu weit ausschweifen und fasste diese kurz zusammen.

Die erste Aufgabe, die ich neben dem eigentlichen Projekt bewältigt habe, war der Übergang vom klassischen package management System PIP zur deutlich automatisierteren Variante Poetry. Poetry ist ein framework, welches automatisiert alle angegebenen Anforderungen (alle Module, die im Projekt benutzt werden) sowie deren transitiven Anforderungen überprüft und installiert. Damit erspart man sich langfristig lästige Kompatibilitätskonflikte und ist in der Lage neuen Entwicklern mit nur einer Zeile Code im Terminal eine neue Arbeitsumgebung zu initialisieren, welche eine virtual environment erschafft und sämtliche Anforderungen installiert und konfiguriert.

Eine weitere Aufgabe war es das sogenannte pre-commit tool mit in das Projekt aufzunehmen und zu konfigurieren. Da größere Softwareunternehmen meist große Entwicklerteams haben, erleichtert es die Arbeit enorm, falls alle Entwickler einheitlichen Code schreiben. Dies wäre zwar recht schwierig umzusetzen, lässt sich aber erreichen, indem man die Entwickler dazu „zwingt“. Damit ist gemeint, dass man gar nicht erst in der Lage ist Code zu committen, falls er nicht eine gewisse Struktur und Syntax erfüllt; hierfür wird pre-commit genutzt.

Üblicherweise arbeiten Softwareunternehmen mit Versionierungstools wie beispielsweise Git. Die Versionierung erfolgt dadurch, dass das aktuelle, funktionierende Projekt auf einem sogenannten „main Branch“ liegt, von dem Entwickler einen neuen, temporären Branch anlegen können, auf dem dann entwickelt wird. Ist die Aufgabe auf diesem temporären Branch erledigt, so wird eine sogenannte GitLab CI/CD Pipeline angeworfen. Die GitLab CI/CD Pipeline ist dabei vereinfacht gesagt eine Reihe von Checks, die der Code erfüllen muss, damit er eingebunden werden kann. Sind alle Checks erfüllt, so kann der temporäre Branch in den main Branch eingebunden werden und wird anschließend gelöscht. Um pre-commit nun auch wirklich zwanghaft in den Workflow mit einzubinden, war meine letzte Aufgabe dieses tool mit in die GitLab CI/CD Pipeline einzubinden, also vereinfacht gesagt pre-commit als ein weiteres Kriterium dieser Pipeline aufzunehmen. Im Zuge dieser Aufgabe strukturierte ich die gesamte Gitlab CI/CD Pipeline um, was eine Folge des Umstiegs von PIP zu Poetry war.

3 Persönliches Fazit

Mein persönliches Fazit zu dieser Werkstudententätigkeit ist, dass ich froh bin viele Aspekte meines Studiums, zumindest die die mit mathematischer Herangehensweise an die Bildgebung und -verarbeitung, statistischem Lernen und der Statistik verwandt sind, wiedergesehen und angewendet zu haben.

Abschließend möchte ich sagen, dass ich in vergleichsweise kurzer Zeit enorm viele Sachen lernen und mitnehmen konnte. Ich bin sehr dankbar dafür die Möglichkeit gehabt zu haben diesen Werkstudentenjob wahrnehmen zu können, da ich mich langfristig auf jeden Fall in der Welt der künstlichen Intelligenz und des maschinellen Lernens sehe. Ich würde jedem, der Interesse an maschinellem Lernen oder Softwareentwicklung im Allgemeinen hat, empfehlen einer vergleichbaren Werkstudententätigkeit nachzugehen. Gerade das Startup-Flair ermöglichte mir mit meinen Kollegen auf Augenhöhe zu arbeiten, wodurch ich nicht nur für die einfachen Aufgaben zuständig war, oder bloß zum zuschauen eingestellt wurde, sondern mich wirklich mitten im Projekt befand und für das Projekt essenzielle Aufgaben übernehmen durfte. Eine solche Verantwortung zu übernehmen wäre vermutlich nicht realistisch in einem größeren Unternehmen.

Stuttgart, 23. Juni 2023