Digital Business University
of Applied Sciences

Cyber- & IT-Security (M.Sc.)

Neuronale Netze & Deep Learning

Sommersemester 2024

Analyse der Gesundheit von Bananenpflanzen durch Bild-Klassifizierung mittels maschineller Lernverfahren

Projektbericht

Eingereicht von:
Elias Häussler

Matrikelnummer:
200094

Dozent:
Tobias Heuser

30. Juli 2024

Inhaltsverzeichnis

Abkürzungsverzeichnis

BananaLSD	Banana Leaf Spot Diseases
CNN	Convolutional Neural Network
ML	Maschinelles Lernen
ReLU	Rectified Linear Unit
RGB	Rot, Grün, Blau

1 Einleitung

1.1 Motivation

Bananen zählen mit einem Anteil von etwa 16% zu den am zweit häufigsten produzierten Früchten der Welt (Mohapatra et al., 2010). Wegen der großen Nachfrage wurden im Laufe der Zeit riesige Bananenplantagen entwickelt. Problematisch wird diese Art des Anbaus immer dann, wenn einzelne Pflanzen erkranken. Verschiedene Krankheitserreger sind besonders gefährlich und müssen frühzeitig erkannt werden, um ein Ausbreiten auf andere Pflanzen zu verhindern (Bhuiyan et al., 2023).

Erkrankungen an Bananenpflanzen lassen sich in der Regel gut identifizieren. Allerdings entwickeln sich mit der Zeit auch immer wieder neue Krankheitserreger, sodass einer frühzeitigen Klassifizierung bekannter und unbekannter Erkrankungen besondere Bedeutung zugesprochen werden mussr (Han et al., 2019). Maschinelles Lernen (ML) kann hier insbesondere in großen, unübersichtlichen Plantagen ein sinnvolles Hilfsmittel sein.

1.2 Herausforderungen

Gut trainierte ML-Modelle können schon bei geringen Anzeichen von Erkrankungen zuverlässige Aussagen über eine mögliche Diagnose treffen und entsprechend frühzeitige Maßnahmen ermöglichen. Die Herausforderung besteht dabei vor allem bei der Klassifizierung neuer oder seltener Krankheiten, denn die Vorhersagen von ML-Modellen sind stets abhängig von den Daten, mit denen sie trainiert wurden. Somit ist es notwendig, ein ML-Modell zu entwickeln, das einerseits zuverlässige Vorhersagen für bekannte Krankheiten bei Bananenpflanzen treffen, andererseits aber auch durch Symbiose mit anderen Pflanzenarten ein global gut funktionierendes Netz an Daten zur Erkennung unbekannter oder seltener Krankheiten aufbauen kann.

2 Initialisierung & Daten

2.1 ML-Problem

Ploetz et al. (2015) beschreiben eindrucksvoll, welche Auswirkungen einzelne Erkrankungen auf das Wachstum von Bananen haben können. Insofern besteht die Aufgabe darin, erkrankte Pflanzen frühzeitig zu erkennen, um eine Ausbreitung zu verhindern. Hierfür bieten sich sinnvolle ML-Maßnahmen an, etwa durch die stetige Beobachtung und Analyse einzelner Pflanzen, was beispielsweise durch die Montage geeigneter Fotokameras erfolgen kann.

Erfahrungen auf dem Gebiet der Erkennung von Krankheiten sind überdies zahlreich vorhanden. Speziell zur Klassifizierung von Bananenpflanzen kann es hilfreich sein, das Expertenteam rund um das Convolutional Neural Network (CNN) *BananaSqueezeNet* in die Konzeptionsphase einzubeziehen. Darüber hinaus sollten Vertreter der beteiligten Unternehmen wie Plantagenbetreiber, Exporteure, Großhändler oder auch Importeure als Stakeholder in den Prozess einbezogen werden. Im größeren Kontext kann es zudem sinnvoll sein, eine staatliche Förderung zu beantragen, um das Projekt erfolgreich bewältigen zu können.

Die Entwicklung eines ML-Modells in diesem Kontext kann sehr ressourcenintensiv sein. Hierzu zählen vor allem die initiale Datenerfassung zur Analyse des ML-Modells sowie die

dafür benötigte Infrastruktur für den Aufbau eines geeigneten Systems. Es muss sichergestellt sein, dass eine fortlaufende Datenerfassung möglich ist, was etwa durch den Einsatz eines Kamerasystems gewährleistet werden kann.

Für den Erfolgsgrad des Systems können aus technischer Sicht unterschiedliche Metriken herangezogen werden, vor allem jedoch die Genauigkeit (*accuracy*) und Empfindlichkeit (*recall*) bei der Erkennung des Gesundheitszustandes. Da es sich bei dem ML-Problem um ein Klassifizierungsproblem handelt, ist es überdies sinnvoll, einen guten Wert bei der Bemessung des *F1-Scores* zu erreichen. Aus betriebswirtschaftlicher Sicht ist vor allem das Erreichen höherer Absatzzahlen bei gleichbleibenden Anbau-Aufwänden eine gute messbare Grundlage für den Erfolg des Projektes. Wenn weniger Bananenpflanzen durch Krankheiten unbrauchbar werden, ist ein betriebswirtschaftlicher Mehrwert zu erkennen.

2.2 Datengrundlage

Eine Erhebung von Daten kranker und gesunder Bananenpflanzen kann je nach Ausprägung sehr schnell zu einem aufwändigen Vorgang werden, weshalb es sich anbietet, auf bestehenden Datengrundlagen aufzubauen, sofern diese qualitativ und quantitativ ausreichend sind. Eine Recherche ergab, dass im Zuge der Entwicklung des CNN *BananaSqueezeNet* der Datensatz *Banana Leaf Spot Diseases (BananaLSD)* mit Fotografien unterschiedlicher Zuständen von Bananenpflanzen erstellt und im Anschluss veröffentlicht wurde (Bhuiyan et al., 2023). Die Fotografien sind in folgende vier Klassen unterteilt:

1. **Cordana** beschreibt Bananenpflanzen, die von dem Krankheitserreger *Cordana musae* befallen sind.
2. **Healthy** beschreibt gesunde Bananenpflanzen, die für ein ausgewogenes Training des ML-Modells dienen.
3. **Sigatoka** beschreibt Bananenpflanzen, die von verschiedenen Krankheitserregern der *Sigatoka*-Krankheit befallen sind.
4. **Pestalotiopsis** beschreibt Bananenpflanzen, die von dem Krankheitserreger *Pestalotiopsis microspora* befallen sind.

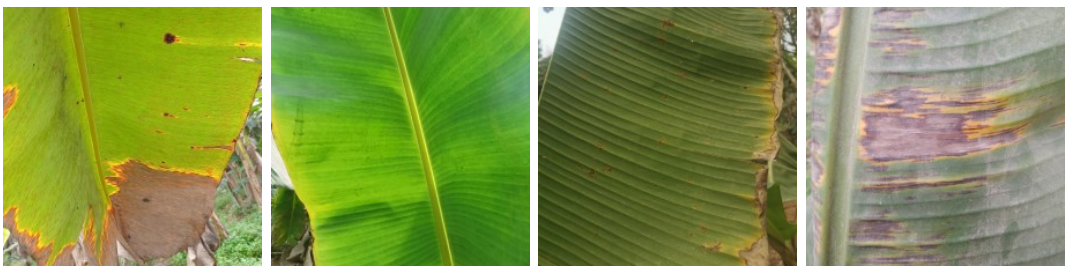

Abbildung 1: Datenpunkte der Klassen *Cordana*, *Healthy*, *Pestalotiopsis* und *Sigatoka*

Die Daten wurden im Juni 2021 in Bangladesch auf einem experimentellen Bananenfeld der *Bangabandhu Sheikh Mujibur Rahman Agricultural University* erhoben (Arman et al., 2023). Alle Fotografien wurden mit Smartphone-Kameras erstellt und im Nachgang von einem

Experten für Pflanzenpathologie mit entsprechenden Labels versehen. Insgesamt enthält der Datensatz 937 Fotografien, wobei den einzelnen Klassen unterschiedlich viele Daten zugrunde liegen. Diesem Klassenungleichgewicht (*class imbalance*) wurde mittels *Data Augmentation* entgegengewirkt. Der veröffentlichte Datensatz enthält sowohl den Roh-Datensatz mit allen 937 Fotografien als auch den erweiterten Datensatz mit jeweils 400 Fotografien pro Klasse (Bhuiyan et al., 2023).

2.3 Datenkorrektur

Für das Training des zu entwickelnden ML-Modells wird der BananaLSD-Datensatz verwendet, wobei bewusst nicht der erweiterte, sondern der Roh-Datensatz Anwendung findet. Im Vergleich zu den Klassifizierungen aus dem Ursprungsdatensatz, bei denen es sich um einen *multi class*-Ansatz handelt, besteht die Aufgabe des neu entwickelten ML-Modells nämlich darin, ein *binary class*-Problem zu lösen, wofür die drei Klassen der erkrankten Bananenpflanzen zusammengefasst werden müssen. Hierbei entstünde bei Nutzung des erweiterten Datensatzes ein größeres Klassenungleichgewicht als bei Nutzung des Roh-Datensatzes. Entsprechend erhalten alle Datenpunkte mit den Bezeichnungen *Cordana*, *Pestalotiopsis* und *Sigatoka* eine neue Bezeichnung *Disease*. Anschließend wird das Problem des Klassenungleichgewichts behandelt. Hierzu wird auf die Datenpunkte beider Klassen (*Healthy* und *Disease*) *Data Augmentation* angewendet, um pro Klasse genau 1000 Datenpunkte zu erhalten.

3 Modellierung

3.1 Transfer Learning

Um eine bestmögliche Performance des ML-Modells zu erreichen, werden zwei vortrainierte Modell im Sinne des *Transfer Learning*-Ansatzes hinzugezogen:

1. **EfficientNet V2** ist eine von Google speziell für die Bild-Klassifizierung und Merkmalsextrahierung entwickelte Kollektion verschiedener ML-Modelle. Es wird in der Variante *Efficientnetv2 B0 (21K)* genutzt, da diese sehr gut auf die Merkmalsextrahierung in der Bild-Klassifizierung und im Speziellen im Kontext von Pflanzen trainiert ist.

2. **rishitdagli/plant-disease** ist ein ML-Modell zur Bild-Klassifizierung erkrankter Pflanzen. Die Nutzung dieses ML-Modells bringt Vorteile insbesondere in der Unterscheidung kranker und gesunder Pflanzenblätter und stellt als spezifischeres ML-Modell eine sinnvolle Ergänzung zu dem globaleren ML-Modell *EfficientNet V2* dar.

3.2 Modell-Architektur

Der Aufbau des ML-Modells ist in unterschiedliche Schichten aufgeteilt:

1. **Input**: In dieser Schicht wird zunächst ein *Tensor* erzeugt, um ein Bild mit 224×224 Pixeln im 3-Kanal-Farbraum (Rot, Grün, Blau (RGB)) abzubilden.

2. **Data Augmentation**: Anschließend wird auf die Eingabe erneut das bereits konfigurierte Verfahren zur Erweiterung der Bildeigenschaften angewandt (siehe Abschnitt 2.3).

3. **Rescaling**: Nun werden die einzelnen Pixel des Eingabe-Bildes entsprechend des im RGB-Farbraum vorhandenen Spektrums von 256 Farben pro Farbkanal skaliert.

4. **Dropout**: Im Sinne der *Regularisierung* und zur Vermeidung von *Overfitting* wird eine *Dropout*-Schicht hinzugefügt. Die *Dropout*-Rate wird zunächst mit `0.5` bestimmt.

5. **Base Models**: Für die Ausgabe werden nun die beiden mittels *Transfer Learning* integrierten ML-Modelle zusammengefügt und als zusätzliche Schicht eingesetzt.

6. **Dense**: Es folgen zwei *Dense*-Schichten:

   (a) Für die erste dieser Schichten werden zunächst 480 Neuronen mithilfe der *Rectified Linear Unit (ReLU)*-Funktion aktiviert.

   (b) Die zweite und zeitgleich letzte Schicht aktiviert ein einzelnes Neuron mithilfe der *Softmax*-Funktion und definiert damit die Ausgabe des ML-Modells. Hierbei wird zusätzlich ein *L2-Regularizer* mit einem Faktor von zunächst `0.001` eingesetzt.

Zusätzlich wird *Early Stopping* auf Basis der Verlustfunktion (*loss function*) eingesetzt, um *Overfitting* zu verhindern. Als Verlustfunktion wird *binary crossentropy* gewählt, um die Genauigkeit bei Vorhersagen des ML-Modells zu überwachen. Das ML-Modell wird zudem mithilfe eines Optimizers kompiliert, um die Gewichte der einzelnen Neuronen möglichst optimal einzustellen. Als Optimizer wird zunächst *Adam* mit einer Lernrate von `0.001` gewählt.

3.3 Performance

Die Performance des ML-Modells kann anhand verschiedener Evaluationsmetriken ausgewertet werden. Zum Einsatz kommen die folgenden Metriken:

- **Accuracy** wird verwendet, um zu messen, wie häufig eine Modellvorhersage korrekt ist. Sie liefert eine Orientierung, wie gut die generelle Performance des ML-Modells ausfällt.
- Der **F1-Score** liefert einen Hinweis auf die Genauigkeit des ML-Modells, indem er das harmonische Mittel von *Precision* und *Recall* berechnet.
- **Precision** gibt an, wie viele der in der Modellvorhersage als „positiv“ ermittelten Daten tatsächlich als „positiv“ zu bewerten sind.
- Ergänzend misst **Recall**, wie viele der tatsächlich als „positiv“ zu bewertenden Daten korrekt als solche erkannt wurden.
- Zudem wird während des Trainings mittels der Verlustfunktion der **Loss** gemessen.

3.4 Training

Für das Training des ML-Modells werden die verschiedenen Komponenten mit jeweils unterschiedlichen Parametern (sog. *Hyperparameter*) konfiguriert und mehrmals hintereinander ausgeführt. Bei jeder Trainingsiteration wird das ML-Modell in maximal 30 Epochen trainiert, wobei im Idealfall zwischen den Epochen eine Verbesserung der einzelnen Metriken erkennbar ist. Zunächst erfolgt ein manuelles Training, gefolgt von einem automaisierten Trainingslauf. Hierfür werden folgende Hyperparameter konfiguriert:

- **Optimizer**: Um die bestmögliche Performance des ML-Modells zu erreichen, ist die Auswahl eines geeiegneten *Optimizers* ein wichtiger Aspekt.
- **Dense-Schichten**: Die Anzahl der Neuronen in der ersten *Dense*-Schicht wird variiert, um die Mustererkennung bei unterschiedlich vielen Neuronen bewerten zu können.
- **Dropout-Rate**: Die Variation der *Dropout*-Rate ermöglicht eine bessere Analyse, inwieweit das ML-Modell auf *Overfitting* reagiert.
- **L2-Faktor**: Auch die unterschiedliche Konfiguration des zusätzlichen *L2-Regularizers* gibt Aufschluss über das Verhalten bei *Overfitting*.
- **Lernrate**: Durch die Konfiguration der Lernrate des eingesetzten *Optimizers* werden Konvergenz und Trainingsgeschwindigkeit des ML-Modells beeinflusst.

4 Ergebnisse

4.1 Manuelles Training

Im manuellen Training werden in vier Iterationen jeweils zufällig erzeugte Hyperparameter gewählt (siehe Tabelle 1).

#	Optimizer	Dense-Units	Dropout-Rate	L2-Faktor	Lernrate
1	Adam	480	0.5	$1 * 10^{-3}$	$1 * 10^{-3}$
2	SGD	192	0.4	$5 * 10^{-4}$	$1 * 10^{-4}$
3	Adam	96	0.2	$2 * 10^{-4}$	$4 * 10^{-4}$
4	RMSprop	32	0.1	$2 * 10^{-5}$	$5 * 10^{-4}$

Tabelle 1: Hyperparameter der manuellen Trainingsiterationen

An den Metriken in Tabelle 2 ist gut erkennbar, dass die gewählten Hyperparameter der zweiten Iteration im Ergebnis am schlechtesten abschneiden. Dies zeigt sich auch dadurch, dass kein *Early Stopping* stattfindet, was im Zusammenhang mit dem hohen *Loss*-Wert auf eine geringe Lern- und gleichzeitig hohe Fehlerrate hindeuten kann. Die Ergebnisse der ersten Iteration sind bereits in Ordnung, können in der dritten Iteration jedoch noch einmal übertroffen werden. Insbesondere der *Loss*-Wert ist in der dritten Iteration im Vergleich zu den ersten beiden Iteration deutlich niedriger. Das Ergebnis in der dritten Iteration deutet insbesondere wegen des frühen *Early Stoppings* auf eine schnelle Konvergenz hin.

#	Accuracy	F1-Score	Precision	Recall	Loss	Early Stopping
1	0.8357	0.8350	0.8410	0.8291	3799	nach 22 Epochen
2	0.6843	0.6926	0.6766	0.7094	0.6012	–
3	0.9014	0.8996	0.9196	0.8803	0.2483	nach 19 Epochen
4	0.9379	0.9379	0.9399	0.9359	0.1565	nach 29 Epochen

Tabelle 2: Evaluationsmetriken der manuellen Trainingsiterationen

Am besten sind jedoch die Hyperparameter der vierten Iteration gewählt. Diese liefert für allen Metriken die jeweils besten Einzelergebnisse und damit die beste Performance. Dies könnte daran liegen, dass durch das Aktivieren weniger Neuronen in der *Dense*-Schicht in

Kombination mit der niedrigeren *Dropout*-Rate das Training weniger komplex ausfällt. Dadurch könnte das ML-Modell möglicherweise besser generalisieren und somit bessere Ergebnisse erzielen, auch weil die Gefahr der *Überregulierung* geringer ist. Das vergleichsweise späte *Early Stopping* deutet zudem auf eine gründliche Optimierung hin.

4.2 Automatisiertes Training

Für das automatisierte Training wird der *Keras Tuner* mit der *Bayesschen Optimierung* eingesetzt. Es werden zehn Trainingsiterationen mit unterschiedlichen Hyperparametern durchgeführt, bei denen das ML-Modell in jeweils maximal 30 Epochen trainiert wird. Die Hyperparameter, die zu den besten Evaluationsmetriken führen, sind in Tabelle 3 abgebildet.

Optimizer	Adam	**Accuracy**	`0.9493`
Dense-Units	`160`	**F1-Score**	`0.9491`
Dropout-Rate	`0.1`	**Precision**	`0.9553`
L2-Faktor	$1 * 10^{-4}$	**Recall**	`0.9430`
Lernrate	$2 * 10^{-3}$	**Loss**	`0.1435`

Tabelle 3: Ergebnisse der besten automatisierten Trainingsiteration

Die besten Ergebnisse aus dem automatisierten Training sind sehr ähnlich zu denen der besten manuellen Trainingsiteration. Es ist erkennbar, dass ein eher niedriger Wert an Neuronen in der *Dense*-Schicht in Kombination mit einer niedrigen *Dropout*-Rate erneut sehr effektiv ist. Alle Metriken sind sehr gut und der *F1-Score* deutet auf ein ausgewogenes Verhältnis von *Precision* und *Recall* hin.

5 Schlussfolgerungen

5.1 Zusammenfassung

Insgesamt fällt das Training und die Performance des ML-Modells sehr gut aus. In der *Confusion-Matrix* spiegeln sich die guten Werte aus den Metriken beiden Trainingsvarianten wider. Damit stellt das ML-Modell für die trainierten Krankheiten an Bananenpflanzen eine solide Grundbasis für die weitere Implementierung auf Plantagen und Ernteanlagen. Es kann bereits in der vorliegenden Form eingesetzt werden, um Ernteausfälle zu verringern und dadurch für einen höheren Absatz zu sorgen.

5.2 Verbesserungspotenziale

Während das ML-Modell für die Erkennung bekannter Krankheiten gut trainiert ist, besteht gleichzeitig ein nicht geringes Risiko, dass neue Krankheiten noch nicht oder nicht zuverlässig erkannt werden. Damit das ML-Modell auch diese Krankheiten zuverlässig erkennt, muss das Training stetig erweitert werden, insbesondere dann, wenn neue Krankheitserreger bekannt werden. Die Architektur des ML-Modells könnte außerdem um weitere vortrainierte ML-Modelle erweitert werden, um zuverlässigere Aussagen treffen zu können. Dabei muss darauf geachtet werden, dass das ML-Modell nicht zu komplex wird, da dies die Performance beeinträchtigen könnte.

Literaturverzeichnis

Arman, S. E., Bhuiyan, M. A. B., Abdullah, H. M., Islam, S., Chowdhury, T. T., & Hossain, M. A. (2023). BananaLSD: A banana leaf images dataset for classification of banana leaf diseases using machine learning. *Data in Brief*, *50*, 109608.

Bhuiyan, M. A. B., Abdullah, H. M., Arman, S. E., Saminur Rahman, S., & Al Mahmud, K. (2023). BananaSqueezeNet: A very fast, lightweight convolutional neural network for the diagnosis of three prominent banana leaf diseases. *Smart agricultural technology*, *4*.

Han, S., Wang, Y., Wang, M., Li, S., Ruan, R., Qiao, T., & Zhu, T. (2019). First report of Pestalotiopsis microspora causing leaf blight disease of Machilus nanmu in China. *Plant Disease*, *103*(11), 2963.

Mohapatra, D., Mishra, S., & Sutar, N. (2010). Banana and its by-product utilisation: An overview. *Journal of scientific and industrial research*, *69*, 323–329.

Ploetz, R. C., Kema, G. H., & Ma, L.-J. (2015). Impact of Diseases on Export and Smallholder Production of Banana. *Annual Review of Phytopathology*, *53*, 269–288.

A Systemanforderungen

Die folgenden Anforderungen an die Systemumgebung sind für die Ausführung des zugehörigen Jupyter Notebooks (`BananaHealthClassification.ipynb`) notwendig:

- Python 3.10
- TensorFlow 2.15

Begründung: In diesem ML-Projekt werden zwei ML-Modelle mittels *TensorFlow Hub* im Sinne des *Transfer Learning*-Ansatzes eingebunden (siehe Abschnitt 3.1). TensorFlow nutzt intern das *Keras*-Framework für verschiedene Programmschritte. Bis *TensorFlow*-Version 2.15 wird *Keras* in Version 2 eingebunden. Mit Veröffentlichung der *TensorFlow*-Version 2.16 wurde die interne Referenz `tf.keras` auf die *Keras*-Version 3 umgestellt[1]. Diese ist zum jetzigen Stand noch nicht kompatibel zu mittels *TensorFlow Hub* geladenen ML-Modellen[2]. Daher kann das vorliegende ML-Projekt aktuell noch nicht mit *TensorFlow*-Version seit 2.16.0 ausgeführt werden. Die daher notwendige *TensorFlow*-Version 2.15 ist wiederum nicht mit neueren *Python*-Versionen kompatibel, weshalb als Anforderung die *Python*-Version 3.10 angegeben ist.

[1] Quelle: Bekanntmachung und Release Notes zur *TensorFlow*-Version 2.16: https://blog.tensorflow.org/2024/03/whats-new-in-tensorflow-216.html (abgerufen am 30. Juli 2024)

[2] Quelle: Issue 903 im `tensorflow/hub`-Repository auf *GitHub*: https://github.com/tensorflow/hub/issues/903 (abgerufen am 30. Juli 2024)

Eigenständigkeitserklärung

Ich trage die Verantwortung für die Qualität des Textes sowie die Auswahl aller Inhalte und habe sichergestellt, dass Informationen und Argumente mit geeigneten wissenschaftlichen Quellen belegt bzw. gestützt werden. Die aus fremden Quellen direkt oder indirekt übernommenen Texte, Gedankengänge, Konzepte, Grafiken usw. in meinen Ausführungen habe ich als solche eindeutig gekennzeichnet und mit vollständigen Verweisen auf die jeweilige Quelle versehen. Alle weiteren Inhalte dieser Arbeit (Textteile, Abbildungen, Tabellen etc.) ohne entsprechende Verweise stammen im urheberrechtlichen Sinn von mir.

Hiermit erkläre ich, dass ich den vorliegenden Projektbericht selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle sinngemäß und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Die vorliegende Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Erklärung zu (gen)KI-Tools

Verwendung von (gen)KI-Tools

Ich versichere, dass ich mich (gen)KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Ich verantworte die Übernahme jeglicher von mir verwendeter Textpassagen vollumfänglich selbst. In der Übersicht verwendeter (gen)KI-Tools habe ich sämtliche eingesetzte (gen)KI-Tools, deren Einsatzform sowie die jeweils betroffenen Teile der Arbeit einzeln aufgeführt. Ich versichere, dass ich keine (gen)KI-Tools verwendet habe, deren Nutzung der Prüfer bzw. die Prüferin explizit schriftlich ausgeschlossen hat.

Hinweis: Sofern die zuständigen Prüfenden bis zum Zeitpunkt der Ausgabe der Aufgabenstellung konkrete (gen)KI-Tools ausdrücklich als nicht anzeige-/kennzeichnungspflichtig benannt haben, müssen diese nicht aufgeführt werden.

Ich erkläre weiterhin, dass ich mich aktiv über die Leistungsfähigkeit und Beschränkungen der unten genannten (gen)KI-Tools informiert habe und überprüft habe, dass die mithilfe der genannten (gen)KI-Tools generierten und von mir übernommenen Inhalte faktisch richtig sind.

Übersicht verwendeter (gen)KI-Tools

Die (gen)KI-Tools habe ich, wie im Folgenden dargestellt, eingesetzt.

(gen)KI-Tool	**Einsatzform**	**Betroffene Teile der Arbeit**
ChatGPT	Unterstützung bei der Fehleranalyse im Entwicklungsprozess	Gesamter Quellcode
	Unterstützung bei der Interpretation von Ergebnissen des manuellen und automatisierten Trainings	Abschnitt 4.1 Abschnitt 4.2

Münster, 30. Juli 2024

Elias Häußler