Duale Hochschule Baden-Württemberg Mannheim

Projektbericht

Car Model Recognition

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

https://github.com/JanMuehlnikel/Car-Model-Recognition

Verfasser:	Jan Mühlnikel, Luca Mohr
Matrikelnummern:	2235021, 7595087
Kurs:	WWI21DSA
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Dozent:	Prof. Dr. Maximilian Scherer
Modul:	Machine Learning Project
Bearbeitungszeitraum:	08.05.2024 – 24.07.2024
Eingereicht:	24.07.2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

KI	Künstliche Intelligenz
ML	Machine Learning
CV	Computer Vision
NN	Neuronales Netzwerk
CNN	Convolutional Neural Network
DL	Deep Learning

1 Einleitung

1.1 Problemstellung

In der modernen Welt gewinnen Technologien wie Künstliche Intelligenz (KI) und Machine Learning (ML) zunehmend an Bedeutung. Besonders im Bereich der Computer Vision wurden in den letzten Jahren erhebliche Fortschritte erzielt. Diese Arbeit befasst sich mit der Anwendung von Computer Vision (CV) Technologien, insbesondere der Bilderkennung. [1]

Ausgangspunkt ist dabei das Problem der Vielzahl von verschiedenen Automodellen. Weltweit gibt es über 110 Automarken, die selbst wiederum eine große Anzahl an verschiedenen Modellen und Modellvarianten anbieten. [2]

Diese Arbeit beschäftigt sich mit dem Problem, dass viele Menschen Autos auf der Straße sehen und sie ansprechend finden, weshalb sie Interesse daran haben, sich ein solches Auto anzuschaffen. Allerdings fällt es vielen schwer, sich die Namen der Hersteller und Modelle zu merken. Diese Herausforderung ist besonders relevant im Kontext von Kaufentscheidungen, bei denen potenzielle Käufer oft von visuellen Eindrücken beeinflusst werden.

1.2 Ziel des Projektes

Das Ziel dieses Projektes ist es, verschiedene Modelle des ML zu trainieren, die in der Lage sind, Automodelle anhand von Fotos zu identifizieren. Diese Modelle sollen im Verlauf dieses Projektes evaluiert und miteinander verglichen werden. Das übergeordnete Ziel besteht darin, das objektiv leistungsfähigste Modell zu identifizieren und in eine mobile Applikation zu integrieren.

2 Hintergrund & Stand der Technik

2.1 Maschinelles Lernen und Bilderkennung

Im folgenden werden die zwei Hauptkonzepte im ML in Bezug auf die Bilderkennung erläutert. Vorausgesetzt für diesen Bericht werden Grundlagen im Bereich ML, Neuronalen Netzwerken und der Evaluierung von ML-Modellen.

2.1.1 CNN

Convolutional Neural Networks (CNN) gehören zu den am häufigsten verwendeten Netzwerken im Bereich des Deep Learning (DL). Insbesondere im Bereich der Computer Vision haben CNN erhebliche Fortschritte ermöglicht. Im Vergleich zu herkömmlichen, vollständig verbundenen neuronalen Netzwerken zeichnen sich CNNs durch einige wichtige Unterschiede aus. In CNNs sind die Verbindungen zwischen Neuronen lokal. Das bedeutet, jedes Neuron in einer Schicht ist nur mit einer begrenzten Anzahl von Neuronen in der vorherigen Schicht verbunden. Dies reduziert die Gesamtanzahl der zu trainierenden Parameter und beschleunigt das Lernen des Netzwerks. Eine weitere Besonderheit ist die Gewichtsteilung in CNNs. Hier teilen Gruppen von Verbindungen dieselben Gewichtungen, was die Anzahl der Parameter weiter minimiert und die Effizienz steigert. Zusätzlich erfolgt in CNNs eine Dimensionsreduktion durch Downsampling, mithilfe von Pooling-Schichten. Dies ermöglicht es, die Größe des Bildes zu verkleinern, die Datenmenge zu reduzieren und gleichzeitig wichtige Informationen zu bewahren, während unwichtige Merkmale entfernt werden.

Im Speziellen der Pooling Layer führt dazu, dass das Problem des Overfittings bei Neuronalen Netzwerken verringert wird, da die Anzahl der Merkmale dadurch verringert werden kann. Für das Pooling werden am häufigsten die max. Pooling und average Pooling Funktionen genutzt. [3]

2.1.2 Transfer Learning

Vortrainierte Modelle in der Bildklassifikation sind Netzwerkarchitekturen, die auf großen Datensätzen mit Millionen von Bildern und Hunderten von Kategorien trainiert wurden. Diese Modelle extrahieren und komprimieren Wissen und Merkmale aus einer Vielzahl von Bildern. Die Stärke vortrainierter Modelle liegt darin, dass sie als Ausgangspunkt für neue Modelle dienen können, die auf kleineren Datensätzen trainiert werden. Durch die Initialisierung eines neuen Modells mit den Gewichtungen und Merkmalen eines vortrainierten Modells kann die Anpassung und Verfeinerung an spezifische Aufgaben oder begrenzte Datensätze schneller und effizienter erfolgen. Dieser Prozess wird als Transfer Learning bezeichnet, bei dem das Wissen aus dem vortrainierten Modell auf das neue Modell übertragen wird. Beispiele für vortrainierte Modelle sind MobileNet und ResNet50. Transfer Learning verbessert die Leistungsfähigkeit und Effizienz neuer Modelle, indem es die bereits gelernten Merkmale und Gewichtungen nutzt. [4]

3 Methoden

3.1 Datensammlung und -vorverarbeitung

3.1.1 Auswahl der Datenquelle

In diesem Projekt wurde entschieden, einen bestehenden Datensatz zu verwenden. Dies geschah aus der Erkenntnis heraus, dass eine eigenständige Datensammlung und -annotation den zeitlichen Rahmen und die projektbezogenen Anforderungen überschreiten würde. Zusätzlich bieten die verfügbaren Online-Datensätze eine umfassende Vielfalt an Automodellen.

Entschieden wurde sich für den Datensatz „DVM-CAR“. Die Eigenschaften des Datensatzes werden in Tabelle 3.1 detailliert dargestellt. Besonders ausschlaggebend für die Auswahl dieses Datensatzes war die große Anzahl an Bildern aus verschiedenen Jahren sowie die Tatsache, dass einige Vorverarbeitungsschritte bereits im Datensatz umgesetzt wurden. Dies erleichtert und beschleunigt das Modelltraining erheblich. In Abbildung 3.1 sind zufällige Beispiele aus dem Datensatz dargestellt. [5]

Tabelle 3.1: Eigenschaften des „DVM-CAR“ Datensatzes

Eigenschaft	**Beschreibung)**
Anzahl der Bilder:	1451784
Anzahl der Automodelle:	899
Ort der Aufnahmen:	UK
Auflösung:	300x300
Hintergrund:	weiß

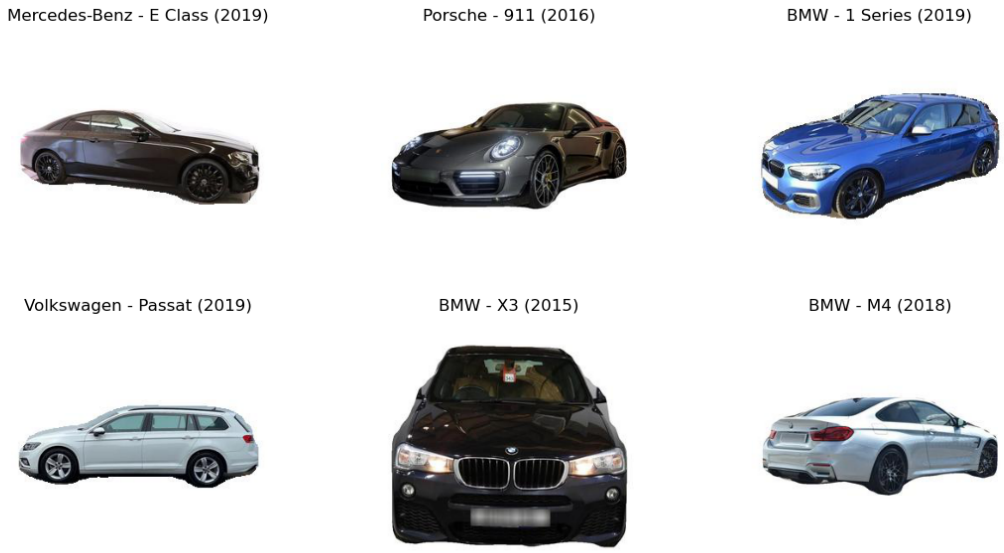

Abbildung 3.1: Zufällige Beispielbilder aus dem DVM-Datensatz. Bilder vorverarbeitet und Hintergrund entfernt.

3.1.2 Vereinfachungen

Aufgrund der begrenzten Ressourcen und Zeit in diesem Projekt wurde der Datensatz angepasst. Dies geschah, um die Trainingszeit zu verkürzen und die Anzahl der vorhergesagten Klassen zu reduzieren. Durch die Reduktion der Klassenanzahl wird die Anzahl der benötigten Beispielbilder verringert, was sowohl die Trainingsdauer erheblich verkürzt als auch den Ressourcenbedarf, insbesondere den Speicherbedarf von RAM und GPU, reduziert. Dieser wird durch die zur verfügungstehenden Ressourcen begrenzt.

Um den Datensatz zu verkleinern, haben wir uns entschieden, uns auf den deutschen Automarkt zu konzentrieren. Dabei liegt der Fokus besonders auf den drei Automarken mit dem größten Marktanteil in Deutschland bis zum Juli 2024. Zusätzlich wird die Marke „Porsche“ einbezogen, um die Dimension der Sportwagen in die Analyse einzubeziehen. [6]

Zusätzlich wurde festgelegt, dass jedes Automodell mindestens 2000 Bilder im Datensatz enthalten muss, um berücksichtigt zu werden. In Abbildung 3.2 sind alle berück-

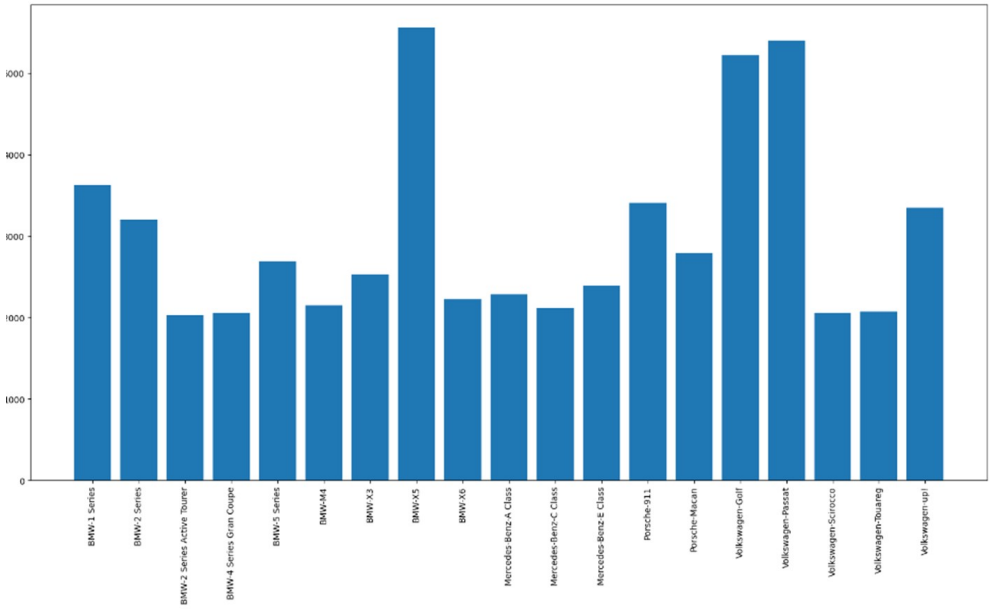

Abbildung 3.2: Die Anzahl der ausgewählten Automodelle im Datensatz erfüllt die Kriterien, dass jedes Modell mehr als 2000 Bilder enthalten muss und Bilder vor dem Jahr 2014 ausgeschlossen wurden.

sichtigten Automodelle mit zusätzlicher Angabe des Vorkommens im Datensatz dargestellt. Darüber hinaus werden Bilder, die Modelle vor dem Jahr 2014 zeigen, nicht einbezogen. Diese Entscheidung wurde getroffen, um zeitliche Dimensionen aus den Modellen zu entfernen und den Trainingsprozess zu vereinfachen. Automodelle können sich im Laufe der Zeit erheblich verändern, was eine differenzierte Betrachtung in verschiedenen Klassen erforderlich macht. Aufgrund begrenzter Ressourcen wurde entschieden, bei den Automodellen keine Unterscheidung nach Jahreszahl vorzunehmen.

3.1.3 Vorverarbeitung

Es wurden keine zusätzlichen Vorverarbeitungsschritte angewendet. Die Daten liegen bereits durch den Datensatz in einheitlicher Größe vor und enthalten entfernten Hintergrund, wodurch weitere Maßnahmen vor dem Modelltraining entfallen.

3.1.4 Aufteilung des Datensatzes

Der Datensatz wurde zufällig in eine Trainingsdatensatz zum trainieren der Modelle, einen Testdatensatz zum evaluieren der Trainsdurchläufe aufgeteilt. Der Split wurde dabei bei 70 % für die Trainingsdaten und 30 % für den Testdatensatz gewählt.

3.2 Modellwahl und -training

3.2.1 Modellwahl

Für das Modelltraining wurden zwei verschiedene Architekturen von neuronalen Netzwerken in Betracht gezogen. Zum einen wurde ein einfaches CNN untersucht, und zum anderen wurde ein Transfer-Learning-Ansatz mit dem ResNet50 verwendet. Diese Auswahl wurde getroffen, um festzustellen, ob ein CNN mit einer einfacheren Struktur und kürzeren Trainingszeiten bereits eine gute Generalisierung auf den Daten erreichen kann oder ob eine komplexere Architektur erforderlich ist. Der Transfer-Learning-Ansatz mit ResNet50 ermöglicht es, von bereits auf großen Datensätzen trainierten Gewichten zu profitieren und diese auf unsere spezifische Aufgabe anzupassen. Diese Methoden stellen moderne Ansätze dar CV Probleme anzugehen. Außerdem wurden zwei verschiedene Varianten der CNN Architektur angeschaut und Parameter angepasst.

Tabelle 3.2: Modellvarianten mit angepassten Parametern

Parameter	**CNN Variante 1**	**CNN Variante 2**	**ResNet50)**
Learning Rate	0.001	LR-Schedule	LR-Schedule
Batchsize	32	16	32
Optimizer	ADAM	ADAM	ADAM
Loss	Cat. Crossentropy	Cat. Crossentropy	Cat. Crossentropy

Tabelle 3.3: Darstellung des Learninrate Schedule

Epoche	Learning Rate
< 5	0.0001
< 10	0.00001
< 15	0.000001
else	0.0000001

3.2.2 Modelltraining

Im folgenden Abschnitt wird das Training der Modelle detailliert dargestellt, wobei besonderes Augenmerk auf die Entwicklung des Loss und der Accuracy der Validierung über die verschiedenen Trainingsperioden gelegt wird. Alle Modelle wurden über insgesamt 14 Epochen trainiert. Ziel dieses Trainingsprozesses ist es, einen kontinuierlichen Lernfortschritt zu erzielen und die Leistung der Modelle stetig zu verbessern.

CNN Variante 1

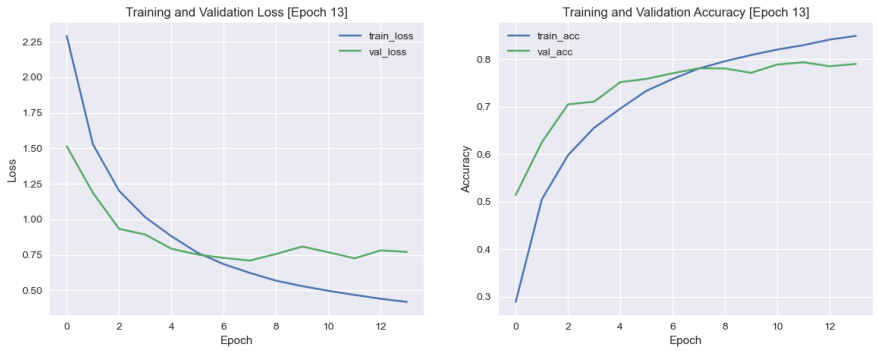

Abbildung 3.3: Darstellung des Validation-Loss und der Validation-Accuracy nach 14 trainierten Episoden des CNN Variante 1 Modells. Ein klarer und stetiger Lernerfolg ist zu verzeichnen. Ergebnisse sind zufriedenstellend.

CNN Variante 2

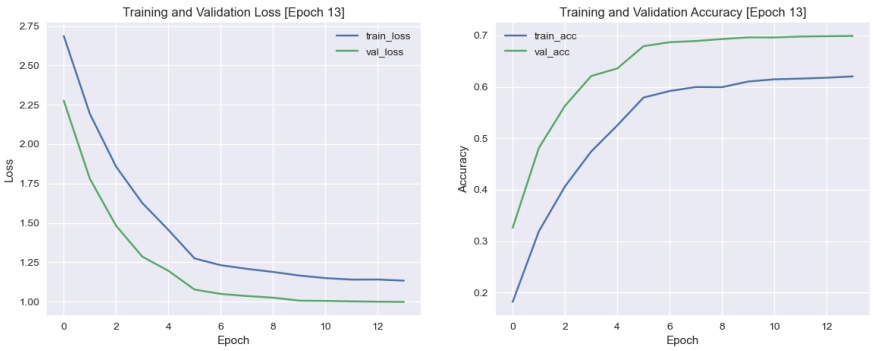

Abbildung 3.4: Die Darstellung des Validation-Loss und der Validation-Accuracy nach 14 trainierten Epochen des CNN Variante 2 Modells zeigt einen klaren und stetigen Lernerfolg. Ab der fünften Epoche ist jedoch kein bedeutender Fortschritt mehr zu erkennen, was darauf hindeutet, dass das Modell ein lokales Minimum erreicht hat. Der Learning Rate Schedule scheint die Fähigkeit des Modells, aus diesem lokalen Optimum herauszukommen, zu beschränken. Es wird empfohlen, eine höhere Lernrate zu verwenden, um das Modell effektiver weiterzuentwickeln. Die Validationsergebnisse zeigen bessere Leistungen als die Trainingsergebnisse, was auf Overfitting hinweist.

ResNet50

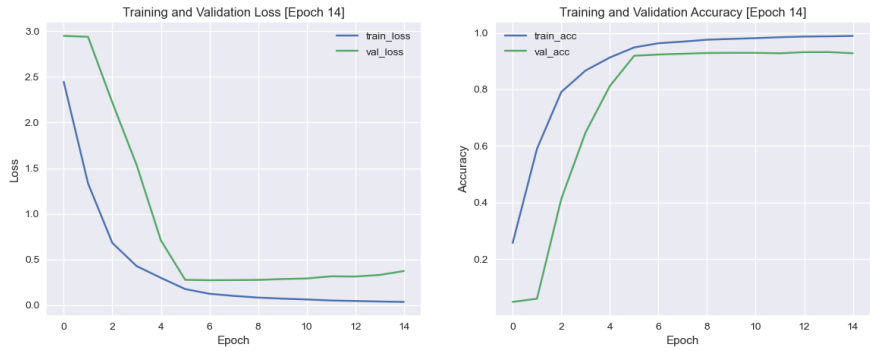

Abbildung 3.5: Darstellung des Validation-Loss und der Validation-Accuracy nach 14 trainierten Episoden des ResNet50 Modells. Es ist zu sehen, dass das Modell schnell auf den Daten generalisiert. Sowohl der Loss sinkt als auch die Accuracy die deutlich bis Periode 6 steigt, zeigen, dass das Modell dazulernt.

3.3 Evaluation & Modellauswahl

Tabelle 3.4 verdeutlicht, dass das ResNet50-Modell die zuverlässigsten Ergebnisse liefert. In allen betrachteten Metriken erzielt es einen beeindruckenden Score von über 0,9 und übertrifft damit die anderen CNN-Modelle signifikant. Daher wurde entschieden, das ResNet50-Modell in der Implementierungsphase in eine Smartphone-App zu integrieren, um von den überlegenen Ergebnissen dieses Modells zu profitieren und eine besonders leistungsfähige und präzise Anwendung zu gewährleisten.

Tabelle 3.4: Evaluationsergebnisse

Metrik	**CNN Variante 1**	**CNN Variante 2**	**ResNet50)**
Accuracy	0.79	0.7	0.93
Precision	0.78	0.7	0.92
Recall	0.77	0.65	0.92
F1-Score	0.77	0.67	0.92

4 Implementierung

4.1 Software und Werkzeuge

In der Entwicklung der Mobile iOS App spielen die Auswahl und der Einsatz der richtigen Software und Werkzeuge eine zentrale Rolle. Diese Technologien unterstützen den Entwicklungsprozess von der Planung über die Programmierung bis hin zum Testing der App. Im Folgenden wird detailliert beschrieben, welche Software und Werkzeuge verwendet wurden.

4.1.1 Entwicklungswerkzeuge

Die eigentliche Programmierung und Entwicklung der App erfolgte mit Hilfe von verschiedenen Entwicklungswerkzeuge:

1. **Xcode:** Xcode ist die integrierte Entwicklungsumgebung von Apple, die speziell für die Entwicklung von iOS-Apps entwickelt wurde. Sie bietet eine Vielzahl von Funktionen, darunter einen leistungsstarken Code-Editor, einen Interface Builder für die Gestaltung der Benutzeroberfläche, Debugging-Tools sowie integrierte Simulatoren. Diese Funktionen ermöglichen es, effizient und produktiv zu arbeiten, während sie Anwendungen erstellen.
2. **Swift:** Die Programmiersprache Swift, die in Xcode verwendet wird, bietet eine Syntax, die für die Entwicklung der App optimiert wurde. Swift ist für seine Sicherheitsfeatures bekannt und eignet sich hervorragend für die Erstellung performanter iOS-Apps.
3. **SF Symbols:** Für die Gestaltung der Benutzeroberfläche wurden SF Symbols verwendet. SF Symbols ist eine Sammlung von über 2.400 konfigurierbaren Symbolen, die von Apple bereitgestellt werden. Diese Symbole sind nahtlos in

das Designsystem von iOS integriert und können leicht an verschiedene Größen und Designs angepasst werden, was eine konsistente und ansprechende Benutzeroberfläche gewährleistet.

4.1.2 Netzwerk- und Tunneling-Tools

Für die Kommunikation zwischen der App und dem Server, der während der Entwicklungszeit als localhost fungierte, wurde ein spezielles Tunneling-Tool verwendet:

1. **ngrok:** Ngrok ist ein Tool, das einen sicheren https-Tunnel zu einem lokalen Server erstellt. Dies ermöglicht es, lokale Anwendungen über das Internet zugänglich zu machen, ohne eine öffentliche IP-Adresse oder eine komplizierte Router-Konfiguration zu benötigen. In diesem Projekt wurde ngrok verwendet, um den Server auf dem localhost für die Bildverarbeitung und Modellvorhersage zugänglich zu machen. Dadurch konnten die Bilder sicher an den Server gesendet und die Vorhersageergebnisse zurück an die App übertragen werden. [7]

4.1.3 Test- und Simulationswerkzeuge

Um die App während der Entwicklungsphase zu testen und sicherzustellen, dass sie auf verschiedenen Geräten und unter verschiedenen Bedingungen einwandfrei funktioniert, wurden Test- und Simulationswerkzeuge eingesetzt:

1. **Xcode Simulator:** Der Xcode Simulator ist ein integraler Bestandteil von Xcode und ermöglicht es, iOS-Apps auf virtuellen Geräten zu testen. Mit dem Simulator können verschiedene iPhone- und iPad-Modelle sowie unterschiedliche iOS-Versionen emuliert werden. Dies hilft dabei, die App auf verschiedene Bildschirmgrößen und Betriebssysteme zu testen, ohne physische Geräte zu benötigen.

4.2 Modellintegration in die App

Der Benutzer öffnet zunächst die App, die auf dem iPhone installiert ist. Nachdem die App gestartet wurde, erhält der Benutzer die Möglichkeit, ein Bild hochzuladen. Hierzu kann er ein bereits vorhandenes Bild aus seiner Fotogalerie auswählen. Sobald das Bild ausgewählt oder aufgenommen wurde, beginnt der Upload-Prozess.

Dieses Bild wird dann anschließend mittels eines sicheren https-Tunnels an den Server geschickt. Ein https-Tunnel sorgt dafür, dass die Daten während der Übertragung verschlüsselt werden.

Der Server, der während der Entwicklung als localhost fungierte, empfängt das hochgeladene Bild. In einer Produktionsumgebung wäre der Server jedoch auf einer tatsächlichen Server-Infrastruktur gehostet, um für alle Benutzer weltweit zugänglich zu sein.

Das auf dem Server befindliche Modell, das speziell für die Erkennung von Automodellen trainiert wurde, bekommt das Bild und analysiert es. Dieses Modell basiert auf fortschrittlichen Algorithmen des maschinellen Lernens und der Bildverarbeitung, die es ihm ermöglichen, das Automodell auf dem Bild zu identifizieren. Der Prozess der Erkennung umfasst mehrere Schritte, darunter die Verarbeitung des Bildes, die Extraktion relevanter Merkmale und die Anwendung des trainierten Modells zur Vorhersage des Automodells.

Nachdem das Modell seine Vorhersage getroffen hat, wird das Ergebnis in einem JSON-Format zurück an die App gesendet. JSON steht für JavaScript Object Notation und ist ein weit verbreitetes Datenformat, das für den Datenaustausch zwischen einem Server und einem Client verwendet wird.

Die App empfängt dann den zurückgesendeten JSON-Wert, der die Vorhersage des Automodells enthält. Um sicherzustellen, dass der Benutzer die Information verstehen kann, übersetzt die App den technischen Wert in eine für den Benutzer verständliche Form. Dies ist die Umwandlung eines numerischen Codes in den Namen des Automodells. Der Benutzer sieht dann das Ergebnis in einer und klar verständlichen Darstellung in der App.

4.3 App-Entwicklung und -Funktionalitäten

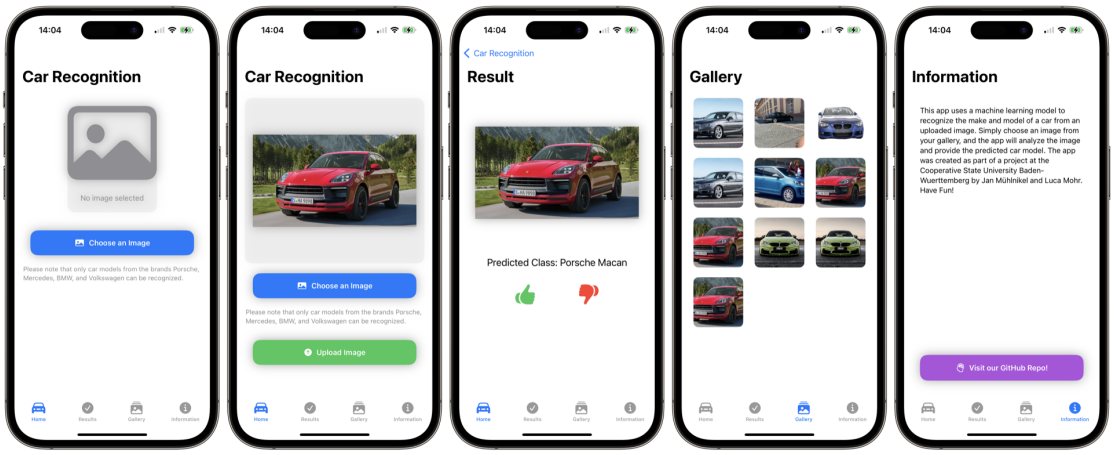

Abbildung 4.1: Hier wird die User Story der App visualisiert.

Die Entwicklung der Mobile iOS App stellt einen umfassenden und detaillierten Prozess dar, der mehrere Phasen durchläuft, um eine benutzerfreundliche und leistungsstarke Anwendung zu gewährleisten. Im Folgenden wird der gesamte Entwicklungsprozess und die damit verbundenen Funktionalitäten der App näher erläutert.

4.3.1 Planungsphase

Zu Beginn der App-Entwicklung steht die Planungsphase, in der die Anforderungen und Ziele der App definiert werden. Hierbei wird festgelegt, welche Kernfunktionen die App bieten soll und welche Plattformen unterstützt werden sollen. Diese Phase ist entscheidend, um sicherzustellen, dass alle Beteiligten ein klares Verständnis von der Vision und den Zielen der App haben.

4.3.2 Designphase

Nach der Planung folgt die Designphase, in der das visuelle und interaktive Design der App erstellt wird. Dies umfasst die Gestaltung der Benutzeroberfläche. Ziel ist es, ein

ansprechendes und intuitives Design zu entwickeln, das den Benutzern eine nahtlose und angenehme Interaktion mit der App ermöglicht.

4.3.3 Entwicklungsphase

In der Entwicklungsphase wird die App tatsächlich programmiert. Hier kommen verschiedene Programmiersprachen und Entwicklungswerkzeuge zum Einsatz, wie Swift für die iOS-Entwicklung und Xcode als integrierte Entwicklungsumgebung. Die geplanten Funktionen sorgen dafür, dass die App stabil und performant läuft. Dabei werden sowohl die Frontend-Komponenten, die für die Benutzer sichtbar sind, als auch die Backend-Services, die im Hintergrund laufen, entwickelt.

4.3.4 Testphase

Nach der Entwicklung folgt eine umfangreiche Testphase. Dabei wird die App auf verschiedenen Geräten und Betriebssystemversionen getestet, um eine optimale Kompatibilität zu gewährleisten.

4.3.5 Bereitstellungsphase

Nachdem die App erfolgreich getestet wurde, folgt die Bereitstellungsphase. Die App wird im Apple App Store eingereicht, wo sie einer Überprüfung unterzogen wird, bevor sie für die Benutzer zum Download bereitgestellt wird. Dieser Prozess stellt sicher, dass die App den Richtlinien von Apple entspricht und keine schädlichen Inhalte enthält.

4.4 Funktionalitäten der App

Die Mobile iOS App bietet eine Vielzahl von Funktionalitäten, die den Benutzern einen hohen Mehrwert bieten. Zu den Kernfunktionen gehören:

1. **Bild-Upload:** Benutzer können Bilder direkt aus ihrer Fotogalerie hochladen.
2. **Bildverarbeitung:** Die App verarbeitet die hochgeladenen Bilder und sendet sie an den Server zur weiteren Analyse.
3. **Automodell-Erkennung:** Der Server analysiert die Bilder und identifiziert das darauf abgebildete Automodell. Die Ergebnisse werden in Echtzeit an die App zurückgesendet.
4. **Benutzerfreundliche Darstellung:** Die App zeigt die erkannten Automodelle in einer klar verständlichen und ansprechenden Weise an, sodass die Benutzer die Informationen leicht verstehen können.
5. **Sicherheitsfeatures:** Durch die Nutzung von https-Tunneln und anderen Sicherheitsmechanismen wird sichergestellt, dass die Daten der Benutzer während der Übertragung geschützt sind.
6. **Updates und Wartung:** Die App wird regelmäßig aktualisiert, um neue Funktionen hinzuzufügen und bestehende zu verbessern. Zudem wird kontinuierlich an der Wartung gearbeitet, um die Stabilität und Sicherheit der App zu gewährleisten.

5 Fazit & Ausblick

Dieses Kapitel beinhaltet die Lessons Learned des Projektes, das Fazit und den Ausblick.

5.1 Lessons Learned

Bei der Entwicklung des Projekts spielte die sorgfältige Auswahl des Datensatzes eine wichtige Rolle. Es musste ein Datensatz ausgewählt werden, der die gewünschten Informationen enthielt und den Anforderungen hinsichtlich Qualität und Umfang entsprach. Die Prüfung und Auswahl stellte sicher, dass die nachfolgenden Schritte effizient durchgeführt werden konnten.

Im nächsten Schritt war die Datenreduktion ein zentraler Aspekt. Da der Datensatz einen sehr großen Umfang hatte, haben wir uns auf die in Deutschland meist verkauftesten Automarken begrenzt.

Ein wichtiger Meilenstein im Projekt war der Einsatz von Transfer Learning unter Verwendung des Modells ResNet50. Dieses vortrainierte Netzwerk ermöglichte es, von bereits vorhandenen und gut verallgemeinerten Merkmalen zu profitieren, was die Effizienz und Genauigkeit des Trainings erheblich steigerte.

Neben der technischen Umsetzung war die Entwicklung einer intuitiven Benutzeroberfläche in der iOS App von großer Bedeutung. Eine benutzerfreundliche Oberfläche stellt sicher, dass die App nicht nur funktional, sondern auch zugänglich und für den Benutzer leicht verständlich ist. Dies erhöht die Nutzung der App und trägt wesentlich zum Erfolg des Projektes bei.

Ein weiterer Erfolg war die Bereitstellung einer funktionierenden Anwendung auf einem Endgerät. Diese praktische Umsetzung hat gezeigt, dass die theoretischen und technischen Ansätze auch in der realen Welt anwendbar und nützlich sind. Entscheidend

war, dass die Anwendung auf den vorgesehenen Geräten zuverlässig und effizient läuft und den Nutzern einen echten Mehrwert bietet.

Nicht zuletzt war die Sicherheit ein zentrales Thema bei der Entwicklung der Anwendung. Der Einsatz von HTTPS war unerlässlich, da Xcode diese als Verbindungsart nutzt und auch erfordert.

5.2 Ausblick

Die nächsten Schritte des Projektes wäre zum einen die Veröffentlichung der App im Apple App Store. Dafür ist die Registrierung eines Apple Developer Accounts notwendig. Außerdem wird die App von Apple selbst überprüft, damit die bereit und nutzbar für die User ist. Ein weiterer Schritt der in diesem Projekt durchgeführt wird, ist die weitere Implementierung von Modellen, um eventuell die Leistung der App als Gesamtkonstrukt zu verbessern.

Literaturverzeichnis

[1] P. Stone, R. Brooks, E. Brynjolfsson, T. Mitchell et al., „Artificial Intelligence and Life in 2030," Stanford University, 2016. Adresse: https://ai100.stanford.edu/sites/g/files/sbiybj18871/files/media/file/ai100report10032016fnl_singles.pdf.

[2] kfz-auskunft.de, *Autohersteller - Alle Automarken und Automodelle auf einen Blick*, Accessed: 2024-07-18, 2024. Adresse: https://www.kfz-auskunft.de/autohersteller.html.

[3] Z. Li, F. Liu, W. Yang, S. Peng und J. Zhou, „A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, Jg. 33, Nr. 12, S. 6999–7019, 2022, ISSN: 2162-237X. DOI: 10.1109/tnnls.2021.3084827.

[4] Valliappa Lakshmanan, Martin Görner, Ryan Gillard, *Practical Machine Learning for Computer Vision*. O'Reilly Media, Inc., 2021. Adresse: https://learning.oreilly.com/library/view/practical-machine-learning/9781098102357/.

[5] D. V. Marketing, *DVM-CAR Dataset*, https://deepvisualmarketing.github.io/, Accessed: 2024-07-18.

[6] Statista, *Monatliche Marktanteile der Automarken in Deutschland*, https://de.statista.com/statistik/daten/studie/235380/umfrage/monatliche-marktanteile-der-automarken-in-deutschland/, Accessed: 2024-07-18.

[7] K. Paulsen, *ngrok Documentation*, Accessed: 2024-07-24, Juli 2024. Adresse: https://ngrok.com/docs/.