Webvisualisierung von Prozesskomponenten in der Bildgebenden Qualitätskontrolle

Studienarbeit 2

des Studienganges Elektrotechnik
an der Dualen Hochschule Baden-Württemberg Mannheim

von
Andreas Braig

02.01.2025

Bearbeitungszeitraum:	07.01.2025 - 07.04.2025
Matrikelnummer, Kurs:	6481829, TEL22AT1
Ausbildungsfirma:	ABB AG
Abteilung:	PAPI-EAM
Betreuer der Dualen Hochschule:	Prof. Dr.-Ing. Bozena Lamek-Creutz

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: „Webvisualisierung von Prozesskomponenten in der Bildgebenden Qualitätskontrolle“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Unterschrift

Vorwort

Die vorliegende Studienarbeit wurde im Rahmen des Studiengangs Elektrotechnik an der DHBW erstellt.

An dieser Stelle möchte ich mich herzlich bei meiner Betreuerin Prof. Dr.-Ing. Bozena Lamek-Creutz und Herrn Shobit Agarwal für ihre wertvolle Unterstützung bedanken.

Zusammenfassung

Abstract

Inhaltsverzeichnis

Inhaltsverzeichnis **V**

Abbildungsverzeichnis **VII**

Listingverzeichnis **VIII**

1. Problemstellung und Ziel dieser Arbeit **1**

2. Theoretische Grundlagen **2**
2.1. Machine Learning und Klassifikationsprobleme 2
2.2. Die Funktionsweise einer API 4
2.2.1. Anfragearten einer API 5
2.2.2. FLASK 6

3. Softwarekonzept **8**
3.1. Programmstruktur 8
3.2. Konfiguration 10
3.3. Die Weboberfläche mittels Python Web API 11

4. Implementierung **12**
4.1. Ablauf der Implementierung 12
4.2. Installation im Labor 12

5. Softwaretests **13**
5.1. Metriken zur Evaluierung 13
5.1.1. Accuracy und Loss 14
5.1.2. Confusion Matrix und F1 Score 15
5.2. Reevaluierung des Modells 16

6. Fazit und Ausblick **17**

Literaturverzeichnis **X**

A. Anhang **XII**

Abbildungsverzeichnis

2.1. Aufbau eines Convolutional Neural Networks [1]. 3
2.2. Schematischer Aufbau der API Kommunikation 4
2.3. Erweiterung der API Darstellung auf Basis von FLASK 6

3.1. Schematische Darstellung der MVC Struktur [12] 8
3.2. Projektstruktur der Software im Dateiverzeichnis 9

5.1. Schematischer Aufbau der API Kommunikation 16

Listings

3.1. Beispiel einer JavaScript Object Notation (JSON)-Datei mit Parametern des mobilnet Modells 10
3.2. Einlesen der JSON-Datei 11

Abkürzungsverzeichnis

CNN Convolutional Neural Network
API Application Programming Interface
PCB Printed Circuit Board
CP-Lab Cyber-Physical Lab
JSON JavaScript Object Notation
REST Representational State Transfer
HTTP Hypertext Transfer Protocol
URL Uniform Resource Locator
HTML Hypertext Markup Language
WSGI Web Server Gateway Interface
MVC Model-View-Controller

1. Problemstellung und Ziel dieser Arbeit

Die zunehmende Automatisierung industrieller Prozesse erfordert zuverlässige Qualitätskontrollsysteme, insbesondere in der Fertigung von elektronischen Baugruppen wie Printed Circuit Boards (PCBs). Im letzten Semester wurde in einer Machbarkeitsstudie untersucht ob ein KI-basiertes System umsetzbar ist, welches mittels TensorFlow Convolutional Neural Networks (CNNs) Defekte auf PCBs erkennt. Dieses System basiert auf einer Online-Implementierung Benutzeroberfläche.

Aktuell bestehen drei zentrale Herausforderungen: Erstens bietet die Online- Implementierung keine lokale Kontrolle über Parameter oder Daten, was die Flexibilität limitiert. Zweitens fehlt eine intuitive Schnittstelle zur Visualisierung von Klassifizierungsergebnissen, was die Benutzerinteraktion erschwert. Drittens soll die Leistung der bisher verwendeten CNN-Architektur evaluiert werden und weitere Architekturen oder Optimierungstechniken sollen getestet werden, um die Erkennungsgenauigkeit zu verbessern.

Ziel dieser Arbeit ist es, die bestehende Lösung in eine lokale Anwendung zu überführen, die folgende Kernkomponenten integriert: Eine zentrale Parametrisierung via JSON, diese ermöglicht die flexible Steuerung aller Modell- und Systemparameter, während eine modularisierte Codebasis die Wartbarkeit und Wiederverwendbarkeit des Python-Codes verbessert. Zusätzlich soll eine Webanwendung mit einer Python Application Programming Interface (API) entwickelt werden, die eine Darstellung in echtzeit von PCBs Klassifizierungsergebnissen bietet. Parallel erfolgt eine systematische Modellre-Evaluation, bei der das aktuelle CNN mit alternativen Architekturen oder Optimierungstechniken verglichen wird.

Durch diese Maßnahmen soll die Darstellung der industriellen Anwendbarkeit im FESTO CP Lab gestärkt werden. Die neue Webvisualisierung soll greifbar machen, was bildverarbeitende Qualitätskontrolle bedeutet, indem sie in Echtzeit Analyse und Ergebnisse der PCBs anschaulich darstellt. Benutzer können direkt sehen, dass Defekte erkannt und klassifiziert werden, was die Transparenz und Nachvollziehbarkeit des gesamten Prozesses erhöht.

2. Theoretische Grundlagen

Zu Beginn dieser Studienarbeit werden die theoretischen Grundlagen erläutert, die für das Verständnis der späteren Kapitel notwendig sind. Dazu gehören die Funktionsweise von Machine Learning und Computer Vision, sowie die Grundlagen einer API. Wie bereits im Ersten Kapitel (siehe Kapitel **??**) beschrieben, baut diese Studienarbeit auf der Arbeit des letzten Semesters auf. Die theoretischen Grundlagen für Machine Learning und Computer Vision werden hier nur kurz erläutert, da sie bereits im letzten Semester ausführlich behandelt wurden.

Die Funktionsweise einer API wird in diesem Kapitel genauer erläutert, da sie eine zentrale Rolle in dieser Studienarbeit spielt. Mithilfe einer Web-API werden die Daten des Python programmes an die entwickelte Webanwendung übertragen.

2.1. Machine Learning und Klassifikationsprobleme

Grundlegend, bevor die Datensätze in Form von Bildern durch Convoliutional Neural Networks CNN analysiert werden können, müssen die Daten verarbeitet werden.

Bildverarbeitung beschäftigt sich mit der Manipulation und Analyse digitaler Bilder durch Algorithmen und bildet die Grundlage für komplexere Verfahren. Ein digitales Bild wird als mehrdimensionale Matrix gespeichert, wobei farbige Bilder als dreidimensionale Tensoren dargestellt werden, deren Dimensionen Höhe, Breite und Farbkanäle repräsentieren. Diese Repräsentation ermöglicht die Anwendung verschiedener Transformationen wie Filterung, Kontrastverbesserung oder geometrische Verzerrungen, die entweder zur Bildverbesserung oder als Vorverarbeitungsschritte für nachfolgende Analysen durch fortschrittlichere Techniken wie Machine Learning und Computer Vision dienen [1].

Convolutional Neural Networks CNN sind spezialisierte Deep Learning Modelle, die für die Verarbeitung von Bilddaten optimiert sind. Sie bilden die Grundlage für zahlreiche moderne Computer Vision Anwendungen, wie Gesichtserkennung und autonome

Fahrzeuge. Die Architektur eines CNN nutzt die räumliche Struktur von Bildern effizient, um visuelle Muster zu erkennen und zu klassifizieren. Ein CNN besteht hauptsächlich aus Faltungsschichten und Linearen Ebenen (siehe Abbildung 2.1). Die Convolutional Layers verwenden kleine Filtermatrizen (Kernel), die über das Bild gleiten und visuelle Merkmale wie Kanten und Texturen erkennen. Diese Filter werden während des Trainings optimiert, um die relevantesten Merkmale zu extrahieren [1] [2].

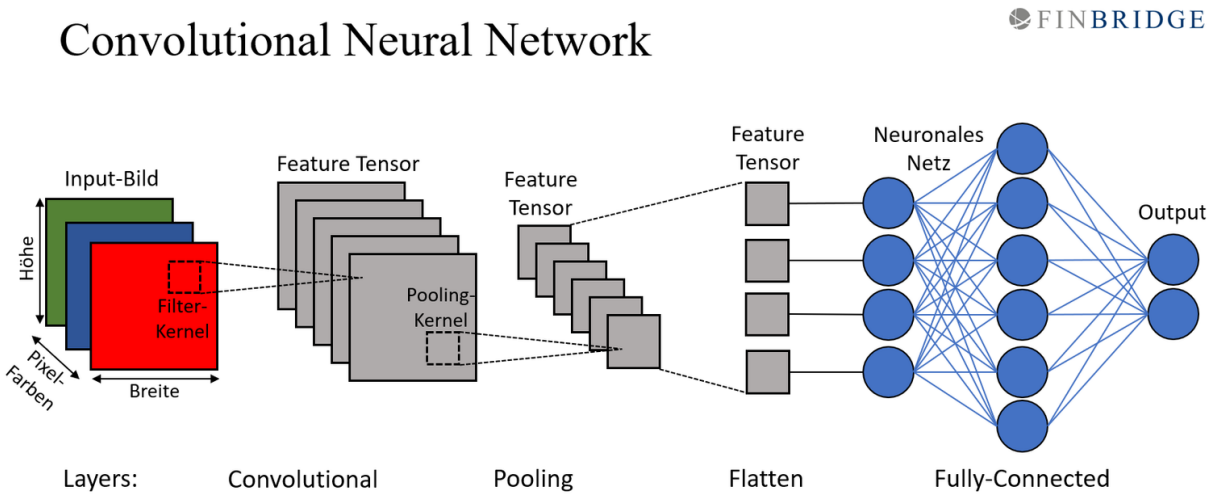

Abbildung 2.1.: Aufbau eines Convolutional Neural Networks [1].

Die Fully-Connected Layers am Ende des Netzwerks (siehe Abbildung 2.1) wandeln die extrahierten Merkmale in eine endgültige Klassifikation oder Vorhersage um. Diese Schichten sind ähnlich wie traditionelle neuronale Netzwerke aufgebaut, wobei jedes Neuron mit allen Neuronen der vorherigen Schicht verbunden ist.

Die wichtigsten Lernansätze im maschinellen Lernen sind überwachtes, unüberwachtes und verstärkendes Lernen. In dieser Arbeit wird überwachtes Lernen genutzt, bei dem ein Algorithmus mit gelabelten Daten trainiert wird, um aus diesen Beispielen zu lernen und Vorhersagen zu treffen.

Beim überwachten Lernen gibt es zwei Hauptmodelle: Klassifikation und Regression. Regression beschreibt kontinuierliche Zusammenhänge zwischen Eingangs- und Ausgangsdaten [3]. Klassifikation teilt Daten in diskrete Gruppen ein. Es sollen keine Kontinuierlichen werte nachgebildet werden. Am Ende des Netzwerks wird mithifge einer $argmax()$ Funktiion eine Klasse fest zugeordnet [4, S. 450]. Die binäre Klassifikation, welche in dieser Arbeit angewandt wird ist eine spezielle Form der Klassifikation, bei der nur zwei Klassen unterschieden werden.

Es gibt verschiedene CNN-Architekturen wie ResNet und MobileNet, die für spezifische Aufgaben besonders gut geeignet sind. Oft werden vortrainierte Modelle verwendet und für spezifische Anwendungsfälle feinabgestimmt, eine Technik bekannt als Transfer Learning, welche auf dem überwachten Lernen basiert. [1].

2.2. Die Funktionsweise einer API

APIs stellen das Herzstück moderner Softwareentwicklung dar und ermöglichen es Programmen, miteinander zu kommunizieren und Daten auszutauschen. Dieser Datenaustausch ist wesentlich für vielfältige Anwendungen, wie etwa das Abrufen von Wetterdaten oder das Interagieren mit sozialen Netzwerken. In der Python Entwicklungsumgebung machen Bibliotheken wie requests oder http.client den Einstieg in die API-Entwicklung besonders zugänglich [5]. Die in dieser Studienarbeit verwendete FLASK Bibliothek ermöglicht es, eine komplextere API zu erstellen, die Daten aus dem Python Programm an die Webanwendung überträgt.

Anwendungsprogrammierschnittstellen API sind Software-Vermittler (Abbildung 2.2) ihre Aufgabe besteht darin, Anwendungen die Kommunikation untereinander zu ermöglichen. Diese subtilen Vermittler sind allgegenwärtig im täglichen Leben, ob bewusst wahrgenommen oder nicht [6].

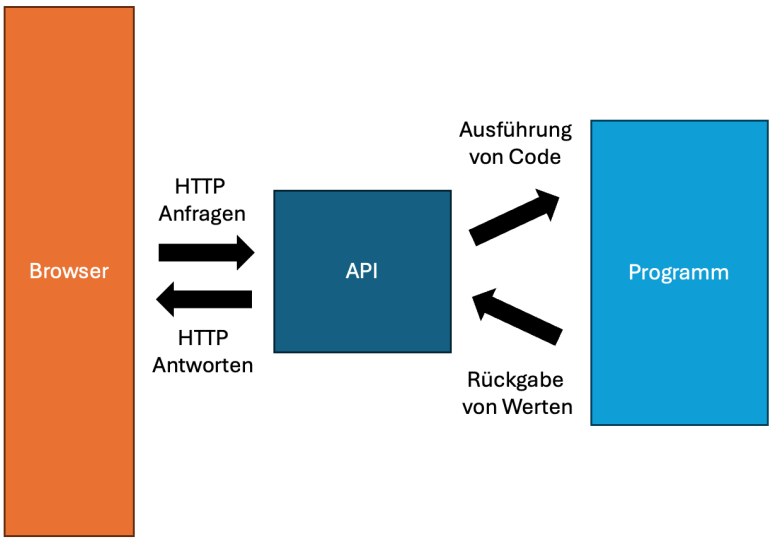

Abbildung 2.2.: Schematischer Aufbau der API Kommunikation

Im Kontext von Webanwendungen bieten APIs den Zugriff auf Daten und Funktionali-

täten von Drittanbietern. Dies ermöglicht es Entwicklern, ihre Anwendungen um Funktionen wie Wetterinformationen, Sportergebnisse, Filmlisten, Tweets, Suchmaschinenergebnisse und Bildverarbeitung zu erweitern. Die Eigenentwicklung solcher Funktionen würde erhebliche Ressourcen beanspruchen, während die Nutzung von APIs eine schnelle und effiziente Integration ermöglicht[7].

2.2.1. Anfragearten einer API

Die Grundbausteine einer API sind Anfragen (Requests) und Antworten (Responses). Diese basieren oft auf dem Hypertext Transfer Protocol (HTTP)-Protokoll, das die Grundlage des Internets bildet [5]. HTTP-Anfragen sind die Art und Weise, wie das Web funktioniert. Jedes Mal, wenn man zu einer Webseite navigiert, stellt der Browser mehrere Anfragen an den Server der Webseite. Der Server antwortet dann mit allen Daten, die für die Darstellung der Seite erforderlich sind, woraufhin der Browser die Seite darstellt [7].

Der generische Prozess der API-Kommunikation lässt sich wie folgt beschreiben: Ein Client, in dieser Studienarbeit der Browser, sendet Daten an eine Uniform Resource Locator Uniform Resource Locator (URL). Der Server unter dieser URL liest die Daten, entscheidet, was damit zu tun ist. Intern wird das passende Python Skript ausgeführt und gibt eine Antwort an den Client zurück. Schließlich verarbeitet der Client die empfangenen Daten entsprechend seiner Programmlogik [7].

Ein wesentlicher Teil der Anfrage ist die Hypertext Transfer Protocol HTTP-Methode. Einige der gebräuchlichsten Methoden sind [8]:

- **GET** Dient dem Abrufen von Daten, ohne Änderungen auf dem Server vorzunehmen
- **POST** Wird verwendet, um neue Daten an den Server zu senden
- **PUT** Aktualisiert vorhandene Daten auf dem Server
- **DELETE** Entfernt Daten vom Server

Diese Methoden bilden die Grundlage des Representational State Transfer Representational

State Transfer (REST)ful API-Designs, das in modernen Webanwendungen weit verbreitet ist [8].

2.2.2. FLASK

Für diese Studienarbeit ist keine Umfängliche WEB-API notwendig. Es soll lediglich eine einzige Hypertext Markup Language (HTML) Datei von der Python Anwendung an den Browser übertragen werden. Es ist für diese Anforderung kein Größeres Framework wie Beispielsweise Django notwendig. FLASK ist ein Mikroframework für Python, das sich auf einfache und schnelle Entwicklung konzentriert. Es ist besonders gut geeignet für kleine bis mittelgroße Projekte, bei denen die Verwendung eines größeren Frameworks übertrieben wäre. FLASK bietet eine Vielzahl von Erweiterungen, die die Entwicklung von Webanwendungen erleichtern. Es ist einfach zu erlernen und bietet eine Vielzahl von Funktionen, die für die Entwicklung von Webanwendungen erforderlich sind.

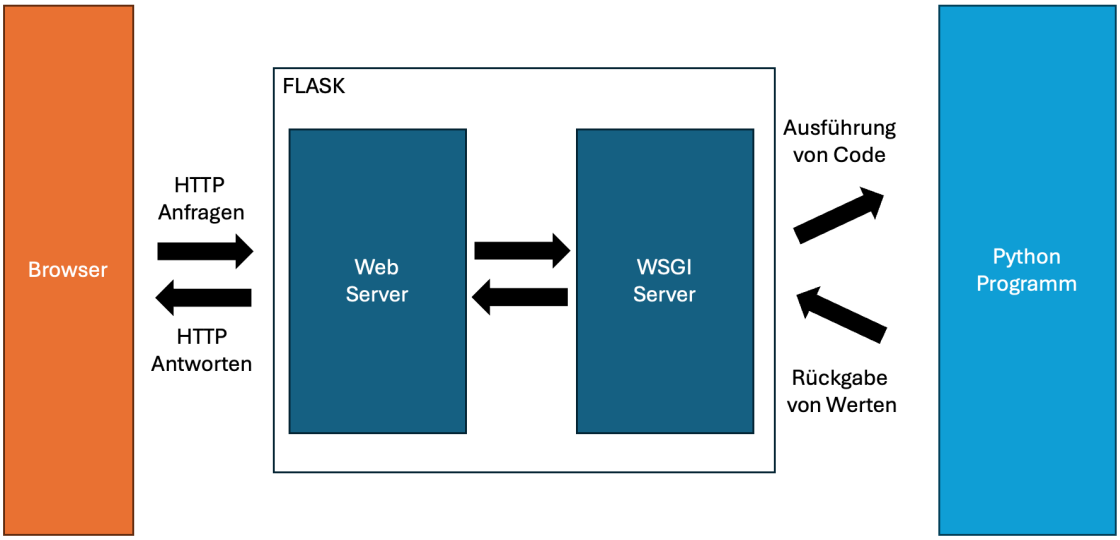

Abbildung 2.3.: Erweiterung der API Darstellung auf Basis von FLASK

FLASK Nutzt die Web Server Gateway Interface (WSGI) Schnittstelle von Python [9] und Abbildung 2.3 und Stellt in Verbindung mit der in dieser Studienarbeit Verwendeten Waitress Bibliothek den Webserver zur Verfügung.

Eine Besondere Anforderung der im Folgenden vorgestellten Webanwendung liegt in der Übertragung der neu erstellten Fotos von der Kamera des FESTO Cyber-Physical Lab (CP-Lab) an die Webanwendung. Um eine einfachere Benutzbarkeit zu gewährleisten soll der Browser nicht jedes mal neu geladen werden müssen, wenn ein Foto geschossen und klassifiziert wird.

Dies geschieht durch die Verwendung von Websockets. Websockets sind eine Technologie, die es ermöglicht, eine bidirektionale Verbindung zwischen einem Client und einem Server herzustellen. FLASK SocketIO ist eine Erweiterung für FLASK, die die Verwendung von Websockets in FLASK Anwendungen ermöglicht und wird in dieser Studienarbeit verwendet, um die Übertragung der Fotos zu realisieren. Auf der gegenseite im HTML Code wird die Bibliothek SocketIO.js verwendet. Diese Basiert auf der Programmiersprache JavaScript und ermöglicht die Kommunikation zwischen dem Browser und dem Server.

Im Folgenden wird die neue Softwarearchitektur vorgestellt und auch die Implementierung der Webanwendung erläutert.

3. Softwarekonzept

Der dieser Studienarbeit zu Grunde liegende Code wurde in der letzten Studienarbeit in einer Cloud-Umgebung entwickelt. Als einfache testumgebung zur umsetzung einer Machbarkeitsstudie war dies ausreichend. Da in dieser Studienarbeit das Umsetzen einer Webvisualisierung der durch die FESTO CP-Lab aufgenommenen Daten im Vordergrund steht, ist es notwendig, die Software in eine lokale Umgebung zu verschieben. Dies ermöglicht eine bessere Kontrolle über die Entwicklungsumgebung und die verwendeten Bibliotheken.

Zusätzlich sollen alle einstellbaren Parameter zentral aufgeführt werden. Dies ermöglicht eine einfache Konfiguration der Software und entspricht dem Stand der Technik[10] [11].

Die Realisierung beider Aufgaben wird in diesem Kapitel beschrieben

3.1. Programmstruktur

Erster Teil der Softwarekonzeption ist der Entwurf der Struktur. Die Struktur einer Software ist entscheidend für die Wartbarkeit und Erweiterbarkeit. Zwei wichtige Kriterien, deren Erhaltung im gesamten Designprozess berücksichtigt werden muss.

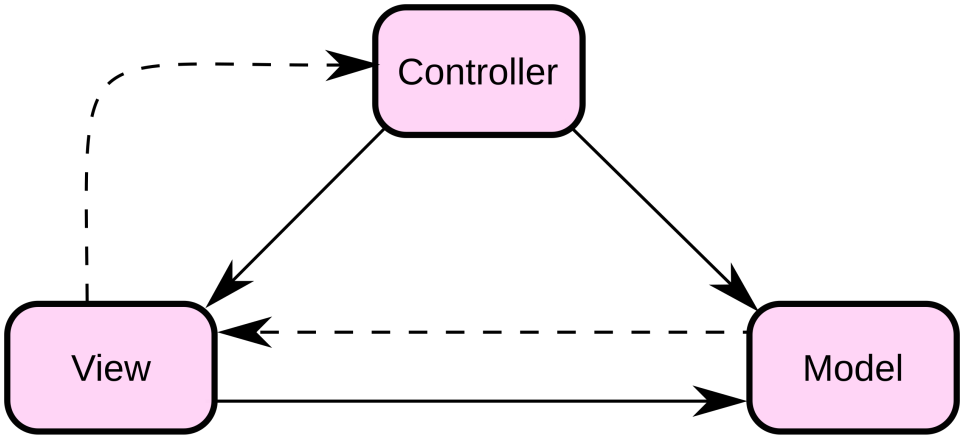

Abbildung 3.1.: Schematische Darstellung der MVC Struktur [12]

Im Rahmen dieser Studienarbeit gibt es keine externen Anforderungen an die Struktur. Daher wurde sich für eine vereinfachte Model-View-Controller (MVC) Struktur entschieden (Abbildung 3.1). Ziel dieser Struktur ist es die Software in drei Teile zu unterteilen. Diese drei Teile sollen eigenständige Aufgaben übernehmen und so verhindern, dass das Programm zu einem monolithischen Codeblock wird.

Der Model-Teil ist für die Datenverarbeitung zuständig und enthält die Datenstrukturen sowie die Logik, die die Daten verarbeitet. Der View-Teil ist für die Darstellung der Daten verantwortlich und umfasst die Benutzeroberfläche sowie die Logik, die die Daten darstellt. Der Controller-Teil steuert die Daten und koordiniert die Kommunikation zwischen Model und View, indem er die entsprechende Logik enthält.

Die vereinfachte Version dieser Struktur kombiniert die Funktionalitäten von View und Controller in der API. (Siehe Abbildung 2.3) und trennt die Datenverarbeitung in einem eigenen Modul, dem Pycore Modul, welches die benötigten selbst entwickelten Bibliotheken zur Verfügung stellt.

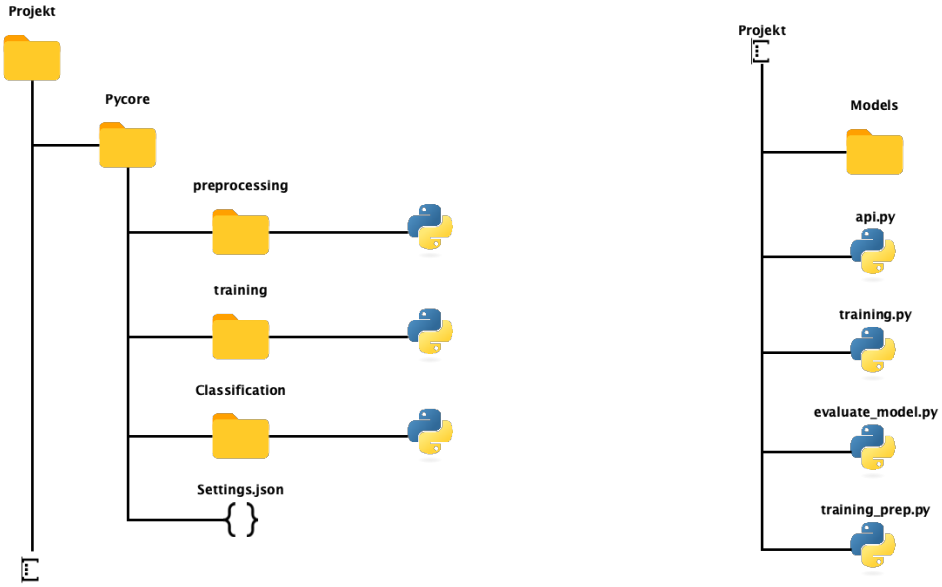

Abbildung 3.2.: Projektstruktur der Software im Dateiverzeichnis

Die in der Grafik Abbildung 3.2 dargestellte Dateistruktur repräsentiert ebenso die Struktur der Software. Die Funktionen werden in Bibliotheken im Pycore-Verzeichnis zusammengefasst. Sämtliche Daten werden in einem ausgegliederten Verzeichnis innerhalb des Projektverzeichnisses abgelegt.

Im Projekthauptverzeichnis gibt es vier Python Skrpte, die ausgeführt werden können, um einzelne Teile der Software auszuführen. `api.py` Wird ausgeführt, um die Weboberfläche zu starten. `train.py` Wird ausgeführt, um das Modell zu trainieren. `evaluate_model.py` Wird ausgeführt, um das Modell zu evaluieren und die in Unterabschnitt 5.1.2. Beschriebenen Werte zu erhalten. `training_prep.py` Kann ausgeführt werden um den Datensatz zu verarbeiten ohne das Modell zu trainieren.

Diese Skripte laden den jeweilig benötigten Programmteil und die benötigten Parameter aus dem Pycore Verzeichnis, führen die entsprechenden Funktionen aus und legen die Ergebnisse im Dateisystem ab.

3.2. Konfiguration

Aus der Aufgabenstellung (siehe Kapitel 1) lässt sich ein weiterer Teil des Softwarekonzepts, die Einstellbarkeit, ableiten. Durch die Einstellbarkeit soll gewährleistet sein, dass die Software flexibel an die Anforderungen des Benutzers angepasst werden kann, ohne dass der Benutzer ein tiefes Verständnis der Software haben muss.

```
{
    "filepaths": {
        "good": "Bilder/Good_Pictures",
        "bad": "Bilder/Bad_Pictures",
        "good_gray": "Bilder/Good_Grayscale",
        "bad_gray": "Bilder/Bad_Grayscale",
        "train": "Bilder/train",
        "test":"Bilder/test",
        "validate":"Bilder/validate",
        "new": "Bilder/new"
    }
}
```

Listing 3.1: Beispiel einer JSON-Datei mit Parametern des mobilnet Modells

Ziel ist es das der Benutzer eine einzige Datei öffnet und dort alle Einstellungen vornehmen kann. Diese Datei soll im JSON-Format vorliegen, da es ein weit verbreitetes Format ist und von vielen Programmiersprachen unterstützt wird [10].

Die Struktur der JSON-Datei ist in Listing 3.1 dargestellt und wird in Pyton mittels des `json` Moduls eingelesen. Ein Beispiel für das Einlesen der Datei ist in Listing 3.2 dargestellt.

Hier wird der Pfad zur Konfigurationsdatei festgelegt und die Datei wird eingelesen. Die Parameter werden dann an die erste Funktion übergeben, welche die Bilder in Graustufen umwandelt und in einem seperaten Verzeichnis ablegt (siehe Listing 3.2).

```
1  import json
2
3  config_path = "pycore/setings.json"
4  cf = json.load(open(config_path, 'r'))
5
6  # Uebergabe der Parameter an die Funktionen
7  uic.folder_to_grayscale(cf["filepaths"]["good"],cf["filepaths"]["good_gray"])
8  uic.folder_to_grayscale(cf["filepaths"]["bad"],cf["filepaths"]["bad_gray"])
```

Listing 3.2: Einlesen der JSON-Datei

Angenommen der Benutzer wünscht ein anderes Verzeichnis für die Bilder, so kann er dies in der JSON-Datei ändern und die Software erneut ausführen, ohne zu wissen, wo die Funktion, welche den Datensatz generiert abliegt.

3.3. Die Weboberfläche mittels Python Web API

4. Implementierung

4.1. Ablauf der Implementierung

4.2. Installation im Labor

5. Softwaretests

In diesem Kapitel wird auf die Evaluierungs- und Testvorgäne für die Sicherstellung der Güte der entwickelten Software eingegangen. Da es sich bei der entwickelten Software um ein Machine-Learning-Modell handelt, wird in diesem Kapitel auf die Evaluierung des Modells eingegangen. Die Funktionen des interpretierten Python Skripts entwickeln keine neuen Algorithmen, die auf ihre Fehleranfälligkeit oder Korrektheit getestet werden müssen.

5.1. Metriken zur Evaluierung

Um zu verstehen wie ein Machine Learning Modell evaluiert werden kann ist es zunächst wichtig die Arten von Kriterien nachzuvollzihen, die für die binäre Klassifikation notwendig sind. Im Folgenden wird allgemein von Instanzen gesprochen, es handelt sich dabei um die Bilder, der PCBs die klassifiziert werden sollen.

- **True Positive (TP)**: Die Anzahl der korrekt klassifizierten positiven Instanzen.
- **True Negative (TN)**: Die Anzahl der korrekt klassifizierten negativen Instanzen.
- **False Positive (FP)**: Die Anzahl der falsch klassifizierten positiven Instanzen.
- **False Negative (FN)**: Die Anzahl der falsch klassifizierten negativen Instanzen.

Diese wesentlichen Typen teilen die Klassifizierten Daten nach dem Test in vier diskrete Kategorien ein. Diese Kategorien bilden die in Unterabschnitt 5.1.2 beschriebene Confusion Matrix. Mit Ihnen können aber auch die Einfacheren Werte Loss und Accuracy berechnet werden. Im Folgenden wird die Mathematik hinter den Methoden vorgestellt.

5.1.1. Accuracy und Loss

Die Accuracy ist eine der einfachsten Metriken zur Evaluierung eines Machine-Learning-Modells. Sie gibt an, wie viele der Instanzen korrekt klassifiziert wurden. Die Accuracy wird vereinfacht wie folgt berechnet:

$$\text{Accuracy} = \frac{n_{\text{richtig klassifiziert}}}{n_{\text{gesamt}}} \tag{5.1}$$

Bezug auf die in Abschnitt 5.1 beschriebenen Kategorien, kann die Accuracy wie folgt berechnet werden [13]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

Die Accuracy wird typischerweise in Prozent angegeben und liegt zwischen 0 und 100%. Sie ermöglicht eine schnelle Einschätzung der Modellgüte. Allerdings kann die Accuracy irreführend sein, da sie die Anzahl der falsch klassifizierten Instanzen nicht berücksichtigt. Ein Modell, das alle Instanzen als negativ klassifiziert, könnte eine hohe Accuracy aufweisen, obwohl es nicht leistungsfähig ist.

Dennoch ist die Accuracy einfacher zu Interpretieren als der hier vorgestellte Loss. Der Loss ist eine Metrik, die die Güte eines Modells anhand der Wahrscheinlichkeiten der Klassifikationen bewertet. Für binäre Klassifikationen wird der Binary Crossentropy Loss verwendet, der wie folgt berechnet wird [13]:

$$\text{Loss} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{5.3}$$

Symbol	Bedeutung
n	Anzahl der Trainingsbeispiele
k	Anzahl der Klassen (bei Mehrklassen-Klassifikation)
y_i	Wahre Klasse (0 oder 1) für das i-te Beispiel
$\hat{y}_i$	Vorhergesagte Wahrscheinlichkeit für Klasse 1

In jeder Epoche des Trainings werden beide Werte berechnet und stellen so die Verbesserung des Modells dar. Der Loss wird dabei minimiert, während die Accuracy maximiert wird. Bereits in der Letzten Studienarbeit wurden die geteste Modelle anhand dieser Metriken evaluiert.

5.1.2. Confusion Matrix und F1 Score

Eine aussagekräftigere Methode zur Evaluierung ist der F1 Score. Um diesen nachzuvollziehen ist zunächst eine aufschlüsselung der Klassifizierten Daten notwendig. Die Confusion Matrix ist eine Tabelle, die die Anzahl der korrekten und falschen Klassifikationen für jede Klasse anzeigt. Die Confusion Matrix hat die folgende Form [14]:

	Vorhergesagt: Positiv	**Vorhergesagt: Negativ**
Tatsächlich Positiv	TP	FN
Tatsächlich Negativ	FP	TN

Hier sind die Werte TP, FP, TN und FN die in Abschnitt 5.1 bereits eingeführt wurden wieder zu finden. Auf der Hauptdiagonale dieser $2x2$ Matrix befinden sich die korrekt klassifizierten Instanzen, während auf der nebendiagonale die falsch klassifizierten Instanzen zu finden sind.

Für jeden Evaluierungsaufruf lässt sich diese Matrix bestimmen. Vorteilhaft an dieser Darstellung ist, dass die Möglichtkeit besteht dieses Modell auf nicht binäre Klassifikation zu erweitern.

Aus dieser Matrix lassen sich nun weitere Metriken ableiten. Eine davon ist der F1 Score. Der F1 Score ist das harmonische Mittel zwischen Precision und Recall. Precision gibt an, wie viele der als positiv klassifizierten Instanzen tatsächlich positiv sind, während Recall angibt, wie viele der tatsächlich positiven Instanzen korrekt klassifiziert wurden. Der F1 Score wird wie folgt berechnet [14]:

$$\text{F1} = \frac{2TP}{2TP + FP + FN} \tag{5.4}$$

Beispielhafte Confusion Matritzen werden in Abschnitt 5.2 vorgestellt.

5.2. Reevaluierung des Modells

In diesem Kapitel soll das Ergebnismodell der letzten Studienarbeit, `"Mobilenet"` erneut Evaluiert werden. Diesmal mithilfe der neuen Metrinken (Unterabschnitt 5.1.2)

Da es sich bei Mobilnet und XX um vortrainierte Modelle von Tensorflow sind, soll auch ein selbstgeschriebenes Modell evaluiert werden. Dieses Modell `"Pytorchmodel"` wurde im Rahmen der Vorlesung Bildverarbeitung entwickelt und soll nun mit in die Tests einfließen.

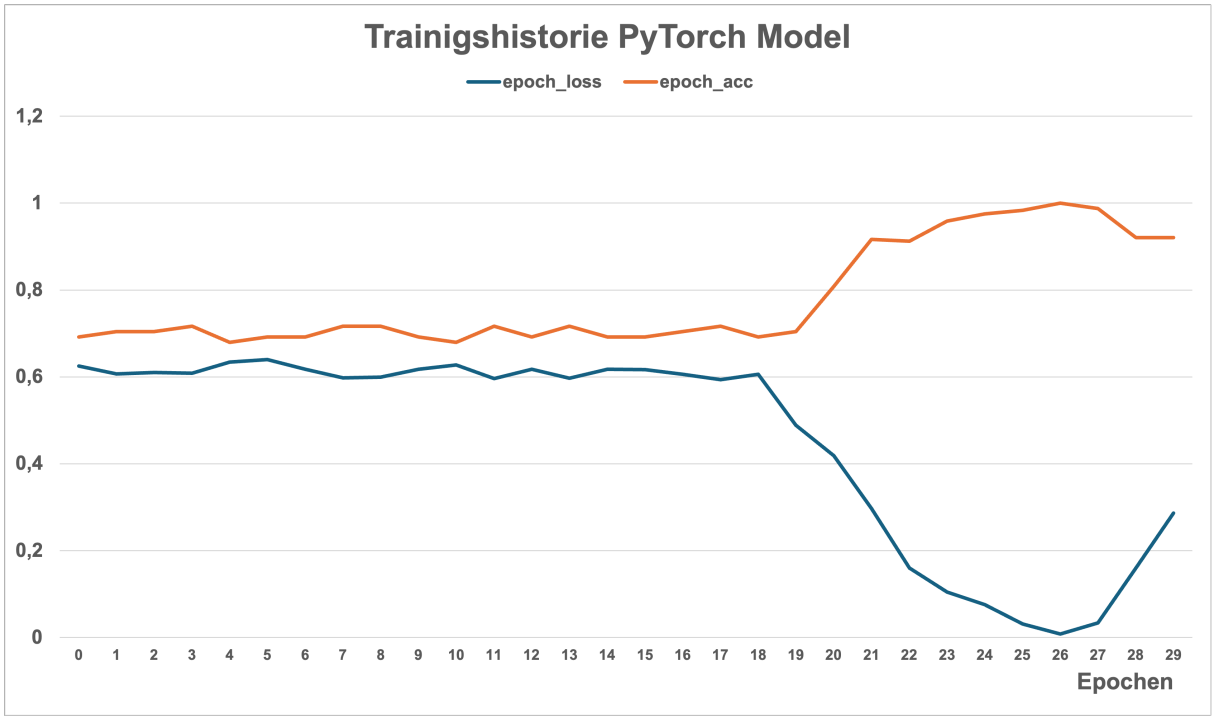

Abbildung 5.1.: Schematischer Aufbau der API Kommunikation

6. Fazit und Ausblick

Literaturverzeichnis

[1] Finbridge.de. „Computer Vision für Finance [Teil 2]: Convolutional Neural Networks," Finbridge GmbH & Co KG. (8. Sep. 2022), Adresse: https://www.finbridge.de/ml-artikel/2022/09/08/computer-vision-fuer-finance-teil-2 (besucht am 01. 03. 2025).

[2] Intel. „Convolutional Neural Networks (CNN) und Deep Learning," Intel. (), Adresse: https://www.intel.com/content/www/de/de/internet-of-things/computer-vision/convolutional-neural-networks.html (besucht am 01. 03. 2025).

[3] „Machine Learning Regression," Mailchimp. (), Adresse: https://mailchimp.com/de/resources/machine-learning-regression/ (besucht am 10. 11. 2024).

[4] H. Süße und E. Rodner, *Bildverarbeitung und Objekterkennung: Computer Vision in Industrie und Medizin*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, ISBN: 978-3-8348-2605-3 978-3-8348-2606-0. DOI: 10.1007/978-3-8348-2606-0. Adresse: https://link.springer.com/10.1007/978-3-8348-2606-0 (besucht am 15. 10. 2024).

[5] S. Mittelstand. „Erste Schritte mit Python HTTP-Anfragen für REST-APIs." (), Adresse: https://www.datacamp.com/tutorial/making-http-requests-in-python (besucht am 01. 03. 2025).

[6] K. Pykes. „Programmieren mit Python APIs: Ihr ultimativer Guide in einfachen Schritten." (), Adresse: https://www.software-mittelstand.info/apis-mit-python-programmieren-eine-schritt-fuer-schritt-anleitung/ (besucht am 01. 03. 2025).

[7] D. O. LLC. „Erste schritte mit der requests-bibliothek in python | DigitalOcean." (), Adresse: https://www.digitalocean.com/community/tutorials/how-to-get-started-with-the-requests-library-in-python-de (besucht am 01. 03. 2025).

[8] C. Rodríguez, M. Baez, F. Daniel u. a., „REST APIs: A large-scale analysis of compliance with principles and best practices," in *Web Engineering*, A. Bozzon, P. Cudre-Maroux und C. Pautasso, Hrsg., Cham: Springer International Publishing, 2016, S. 21–39, ISBN: 978-3-319-38791-8. DOI: 10.1007/978-3-319-38791-8_2.

[9] Flask. „Welcome to Flask — Flask Documentation (3.1.x).“ (), Adresse: https://flask.palletsprojects.com/en/stable/ (besucht am 01. 03. 2025).

[10] *Diskussion über die Projektstrukturen von Python ML Projekten*, unter Mitarb. von M. B. Gür, 8. Nov. 2024.

[11] P. Oliveira. „How to write a python configuration file,“ LambdaTest. Section: Selenium Python. (29. Sep. 2023), Adresse: https://www.lambdatest.com/blog/python-configuration-file/ (besucht am 02. 03. 2025).

[12] *Model View Controller*, in *Wikipedia*, Page Version ID: 248816402, 22. Sep. 2024. Adresse: https://de.wikipedia.org/w/index.php?title=Model_View_Controller&oldid=248816402 (besucht am 02. 03. 2025).

[13] A. Wiki. „Accuracy and loss | AI wiki.“ (17. Dez. 2019), Adresse: https://machine-learning.paperspace.com/wiki/accuracy-and-loss (besucht am 02. 03. 2025).

[14] Z. C. Lipton, C. Elkan und B. Narayanaswamy, *Thresholding Classifiers to Maximize F1 Score*, 14. Mai 2014. DOI: 10.48550/arXiv.1402.1892. arXiv: 1402.1892[stat]. Adresse: http://arxiv.org/abs/1402.1892 (besucht am 02. 03. 2025).

A. Anhang