



Technische  
Hochschule  
Nürnberg

Fakultät Informatik

**Vergleichsanalyse von multimodalen Large  
Language Models und einer  
OCR/YOLO-Pipeline zur  
Dokumentenklassifikation für die  
Bundesagentur für Arbeit**

Bachelorarbeit im Studiengang Informatik

vorgelegt von

Lukas Müller

Matrikelnummer 3698673

Erstgutachter: Prof. Dr. Natalie Kiesler

Zweitgutachter: Prof. Dr. Korbinian Riedhammer

© 2026

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

## **Kurzdarstellung**

Kurze Zusammenfassung der Arbeit, höchstens halbe Seite. Deutsche Fassung auch nötig, wenn die Arbeit auf Englisch angefertigt wird.



# Inhaltsverzeichnis

<b>1 Einleitung</b>	.	.	.	.	.	.	.	.	.	.	<b>1</b>
1.1 Motivation	.	.	.	.	.	.	.	.	.	.	<b>1</b>
1.2 Problemstellung	.	.	.	.	.	.	.	.	.	.	<b>1</b>
1.3 Zielsetzung	.	.	.	.	.	.	.	.	.	.	<b>2</b>
<b>2 Theoretische Grundlagen</b>	.	.	.	.	.	.	.	.	.	.	<b>3</b>
2.1 Dokumentenarten	.	.	.	.	.	.	.	.	.	.	<b>3</b>
2.1.1 KG5b	.	.	.	.	.	.	.	.	.	.	<b>3</b>
2.1.2 Ausbildungsvertrag	.	.	.	.	.	.	.	.	.	.	<b>3</b>
2.1.3 Sonstige Dokumente	.	.	.	.	.	.	.	.	.	.	<b>4</b>
2.2 Fachliche Grundlagen	.	.	.	.	.	.	.	.	.	.	<b>4</b>
2.2.1 Der Ist-Zustand	.	.	.	.	.	.	.	.	.	.	<b>4</b>
2.2.2 Rahmenbedingungen und Infrastruktur	.	.	.	.	.	.	.	.	.	.	<b>6</b>
2.3 Vision Language Models (VLMs)	.	.	.	.	.	.	.	.	.	.	<b>7</b>
2.4 Vorstellung der Modelle	.	.	.	.	.	.	.	.	.	.	<b>8</b>
2.4.1 Pixtral-12B	.	.	.	.	.	.	.	.	.	.	<b>8</b>
2.4.2 Qwen-2.5-VL	.	.	.	.	.	.	.	.	.	.	<b>9</b>
2.4.3 Benchmarks	.	.	.	.	.	.	.	.	.	.	<b>9</b>
2.5 Parameter-Efficient Fine-Tuning	.	.	.	.	.	.	.	.	.	.	<b>9</b>
2.6 Verwandte Arbeiten	.	.	.	.	.	.	.	.	.	.	<b>10</b>
<b>3 Methodik und Implementierung</b>	.	.	.	.	.	.	.	.	.	.	<b>11</b>
3.1 Modellierung des Dokumentenbestands als Grundlage der Evaluation	.	.	.	.	.	.	.	.	.	.	<b>12</b>
3.1.1 Preprocessing des Dokumentenbestands	.	.	.	.	.	.	.	.	.	.	<b>12</b>
3.1.2 Modellierung der Dokumentenarten als JSON-Objekte	.	.	.	.	.	.	.	.	.	.	<b>13</b>
3.1.3 Test- und Validierungsdatensatz	.	.	.	.	.	.	.	.	.	.	<b>15</b>
3.2 Evaluation der generierten JSON-Objekte	.	.	.	.	.	.	.	.	.	.	<b>15</b>
3.2.1 Validierung des VLM-Outputs	.	.	.	.	.	.	.	.	.	.	<b>15</b>
3.2.2 Vergleichslogik der unterschiedlichen JSON-Felder	.	.	.	.	.	.	.	.	.	.	<b>15</b>
3.2.3 Bewertung der Klassifikation und der Information Extraction	.	.	.	.	.	.	.	.	.	.	<b>17</b>
3.3 Messung von Latenz, Energieverbrauch und VRAM-Nutzung	.	.	.	.	.	.	.	.	.	.	<b>19</b>
3.4 Auswahl des Basismodells	.	.	.	.	.	.	.	.	.	.	<b>20</b>
3.5 Fine-Tuning des Basismodells	.	.	.	.	.	.	.	.	.	.	<b>21</b>
3.5.1 Trainingsdatensatz	.	.	.	.	.	.	.	.	.	.	<b>21</b>

3.5.2 Optimierung des Fine-Tunings . . . . .	22
<b>4 Ergebnisse . . . . .</b>	<b>25</b>
4.1 Ergebnisse der YOLO/OCR-Pipeline gegenüber der Basismodelle . . . . .	25
4.2 Ergebnisse des weitertrainierten Modells gegenüber dem größeren Basismodell . .	27
4.3 Ressourcenverbrauch der VLMs gegenüber der YOLO/OCR-Pipeline . . . . .	29
<b>5 Diskussion . . . . .</b>	<b>33</b>
<b>6 Zusammenfassung . . . . .</b>	<b>35</b>
<b>Abbildungsverzeichnis . . . . .</b>	<b>37</b>
<b>Tabellenverzeichnis . . . . .</b>	<b>39</b>
<b>Listingverzeichnis . . . . .</b>	<b>41</b>
<b>Literaturverzeichnis . . . . .</b>	<b>43</b>
<b>Glossar . . . . .</b>	<b>45</b>

# Kapitel 1

## Einleitung

### 1.1 Motivation

Die öffentliche Verwaltung in Deutschland steht vor einer großen Herausforderung. Durch den demografischen Wandel verliert der öffentliche Sektor eine hohe Anzahl an erfahrenen Sachbearbeitern, während der Anspruch der Bürger steigt. Dieser Wandel wird ohne Digitalisierung und Automatisierung der Prozesse kaum abzufangen sein.

Auch in der Bundesagentur für Arbeit (BA) macht sich dieser Wandel, hin zu mehr Digitalisierung, bemerkbar. Ein Beispiel hierfür ist die Bearbeitung von Kindergeldanträgen. Mit einem jährlichen Aufkommen von mehreren Millionen Anträgen[Arbe 24] und saisonalen Spitzen, kommen hier die Sachbearbeiter an ihre Grenzen.

Um sich dieser Herausforderung zu stellen, wurde bereits im Oktober 2025 ein teilautomatisiertes System produktiv gesetzt. Hierbei wird sich auf die Kindergeldanträge von volljährigen Auszubildenden konzentriert. Es werden Dokumente klassifiziert und mithilfe von YOLO- und OCR-Modellen Inhalte aus hochgeladenen Dokumenten erkannt und dem Sachbearbeiter Vorschläge angezeigt. Diese aktuelle YOLO/OCR-Pipeline wertet die Bearbeitung der Kindergeldanträge für die Sachbearbeiter bereits auf, stößt jedoch an ihre Grenzen.

Aufgrund der rasanten Entwicklung im Bereich der Vision Language Models (VLMs) werden diese zunehmend interessanter. In der vorliegenden Arbeit soll evaluiert werden, inwiefern ein solches Modell Potenzial hat, die YOLO/OCR-Pipeline zu ersetzen.

### 1.2 Problemstellung

Seitdem die aktuelle Pipeline im operativen Einsatz ist, zeigen sich verschiedene Herausforderungen.

Das Hauptproblem ist die hohe Varianz der hochgeladenen Dokumente, insbesondere der Ausbildungsverträge. Die verschiedenen Firmen und Kammern nutzen alle unterschiedliche

Layouts, wodurch das Training der Modelle komplex ist. Es ist schwierig, die Varianz in den Trainingsdaten abzubilden, zudem es zeitaufwendig ist, eine solche Menge an Trainingsdaten bereitzustellen.

Ein weiteres Problem ist die Qualität der hochgeladenen Dokumente. Die Dokumente, die relevant für den Kindergeldantrag bei volljährigen Auszubildenden sind, benötigen in allen Fällen eine Unterschrift. Dadurch werden die Dokumente zwingenderweise ausgedruckt, ausgefüllt und unterschrieben, um schlussendlich wieder eingescannt zu werden. Dieser Prozess, den man auch Medienbruch nennt, hat eine schlechte Qualität zur Folge.

Angesichts der Verwendung von vielen unterschiedlichen Modellen ist die Wartung des Systems aufwändig. Eine Anpassung an neue Gegebenheiten ist ein zeitaufwendiger Prozess. Schon kleine Änderungen an der Laufzeitumgebung, wie zum Beispiel eine neue Python-Version, erfordern meistens ein erneutes Training der Modelle.

### 1.3 Zielsetzung

Das Ziel dieser Bachelorarbeit ist die Entwicklung und Evaluation eines prototypischen Systems zur teilautomatisierten Klassifikation und Informationsextraktion von Dokumenten. Angesichts der hohen Anzahl manuell bearbeiteter Anträge soll untersucht werden, inwieweit VLMs diesen Prozess effizienter gestalten können.

Es wird eine Pipeline implementiert, die in der Lage ist, gescannte Dokumente als Bilddaten zu verarbeiten. Das System soll den Dokumententyp eigenständig erkennen und definierte Inhalte, darunter handschriftliche Merkmale wie Unterschriften oder Stempel, in ein standardisiertes Format überführen.

Ein weiteres Ziel ist der Vergleich unterschiedlicher Modellansätze hinsichtlich ihrer Extraktionsperformance und Effizienz. Hierbei wird ein kleineres, domänenpezifisch nachtrainiertes Modell gegen leistungsstärkere Modelle mit höherer Parameteranzahl antreten. Es soll ermittelt werden, ob durch Fine-Tuning mit einem begrenzten Datensatz vergleichbare oder bessere Ergebnisse erzielt werden können als durch den Einsatz größerer Basismodelle.

Diese Arbeit konzentriert sich auf die Machbarkeit und die Evaluation der Modelle anhand eines Testdatensatzes. Die Entwicklung zielt auf einen funktionsfähigen Prototyp ab, der lokal betrieben wird. Eine vollständige Integration in das bestehende operative Fachverfahren ist nicht Gegenstand dieser Arbeit.

# Kapitel 2

## Theoretische Grundlagen

### 2.1 Dokumentarten

Im Rahmen der Kindergeldbeantragung für Auszubildende sind verschiedene Nachweise gültig. Zu den anerkannten Dokumententypen zählen der offizielle Vordruck der Bundesagentur für Arbeit (KG5b)[[Arbe22](#)] sowie Ausbildungsverträge. Die relevanten Informationen aus diesen Dokumenten sind von der fachlichen Seite vorgegeben. Zusätzlich laden Kunden häufig weitere Unterlagen, wie beispielsweise Schulbescheinigungen, im Portal hoch. Da diese für den Kindergeldantrag nicht im Fokus stehen, werden sie im Folgenden unter der Kategorie „Sonstiges“ zusammengefasst.

#### 2.1.1 KG5b

Das Formular KG5b ist, wie bereits erwähnt, ein offizielles Dokument der Bundesagentur für Arbeit, welches als Bescheinigung der Ausbildungsstätte dient. Volljährige Kinder weisen damit gegenüber der Familienkasse den Status ihrer Ausbildung nach, was die Voraussetzung für den weiteren Kindergeldbezug ist.

Abbildung [2.1](#) zeigt ein exemplarisch ausgefülltes KG5b-Formular mit Markierung der für den Kindergeldantrag relevanten Felder.

#### 2.1.2 Ausbildungsvertrag

Im Gegensatz zu den standardisierten KG5b-Formularen weisen Ausbildungsverträge eine deutlich höhere Varianz auf. Dies ist auf die Vielzahl unterschiedlicher zuständiger Stellen (z. B. Industrie- und Handelskammern, Handwerkskammern, Ärztekammern) und Firmen zurückzuführen, die jeweils ein eigenes Layout definieren. Die Vielfalt der Dokumentenstruktur reicht dabei von formularbasierten Layouts bis hin zu unstrukturierten Fließtexten.

In der folgenden Abbildung [2.2](#) ist ein synthetischer Ausbildungsvertrag der Industrie- und Handelskammer[[Hand25](#)] abgebildet, in dem die relevanten Felder markiert sind.

**Erklärung zum Ausbildungsverhältnis**

**Angaben zum Kind**

**Angaben zum Ausbildungsverhältnis**

**Hinweis an den Kindergeldberechtigten: Bitte füllen Sie Punkt e) erst aus, nachdem der Ausbildungsbetrieb die Angaben zum Ausbildungsverhältnis bestätigt hat!**

**Wir versichern, dass unsere Angaben vollständig sind und der Wahrheit entsprechen. Uns ist bekannt, dass wir alle Daten, die für den Anspruch auf Kindergeld von Bedeutung sind, unverzüglich der Familienkasse mitzuteilen haben. Den Inhalt des Merkblattes Kindergeld (zu finden unter www.bzg.de oder www.familienkasse.de) haben wir zur Kenntnis genommen.**

**Hinweise zum Datenschutz:** Ihre Daten werden gemäß der §§ 31, 92 bis 70 Einkommensteuergesetz und der Regelungen der Abgabenordnung bzw. aufgrund des Bundeskindergeldgesetzes und des Sozialgesetzbuches verarbeitet. Zweck der Verarbeitung der Daten ist die Prüfung Ihres Anspruchs auf Kindergeld. Nähere Informationen über die Verarbeitung Ihrer Daten durch die Familienkasse und zu Ihren Rechten nach Artikel 13 bis 22 der Datenschutz-Grundverordnung erhalten Sie im Internet auf der Seite [www.familienkasse.de](#) (zu finden unter [www.arbeitsagentur.de/datenbehandlung](#)). Weitere Informationen über die Kontaktstellen der Kreisdatenschutzbeauftragten befinden sich auf der Internetseite [www.kreisdatenschutz.de](#). Die Daten der Kindergeldanträge werden in der Regel nach dem Ende der Kindergeldzahlung noch für 6 Jahre aufbewahrt.

**Unterschrift der kindergeldberechtigten Person bzw. der gesetzlichen Vertretung**

**Unterschrift des vorjährigen Kindes**

**All Eingaben löschen** **Drucken** **Speichern**

(a) Erste Seite

(b) Zweite Seite

Abbildung 2.1: KG5b-Formular

### 2.1.3 Sonstige Dokumente

Die Kategorie **Sonstiges** steht als Auffangklasse für alle restlichen Dokumente bereit. Darin befinden sich zum Beispiel Schulbescheinigungen, Anträge auf Eintragung bei der Handelskammer oder Studienbescheinigungen. Da diese Dokumente keine Relevanz für die Weiterbeantragung des Kindergeldes bei volljährigen Auszubildenden haben, werden keine Informationen aus ihnen benötigt.

## 2.2 Fachliche Grundlagen

### 2.2.1 Der Ist-Zustand

Der schematische Ablauf der YOLO/OCR-Pipeline ist in Abbildung 2.3 dargestellt.

Der Prozess der Dokumentenverarbeitung lässt sich in zwei Stufen einteilen.

In der ersten Stufe werden die hochgeladenen Dokumente des Kunden entgegengenommen. Wenn ein Dokument als PDF vorliegt, wird mithilfe des Textlayers und eines OCR-Modells

**Berufsausbildungsvertrag**  
§§ 10, 11 des Berufsbildungsgesetzes – BBG

Zwischen dem Ausbildenden (Ausbildungsbetrieb) und dem Auszubildenden wird nachstehender Berufsausbildungsvertrag zur Ausbildung im Ausbildungsbetrieb

Apotheker  
(wenn einschlägig, bitte einschließlich Fachrichtung, Schwerpunkt, Wahlqualifikationen) und/oder Einsatzgeber nach der Ausbildungsvorordnung bestimmt

nach Maßgabe der Ausbildungsordnung<sup>1</sup> geschlossen.

Beispielschule  
Zuständige Berufskammer  
Änderungen an den vertraglichen und von Auszubildenden unterschriebenen Eintragungen in den Verträgen der Betriebsvereinbarungen gelten für die Berufsausbildung ab dem Zeitpunkt der Änderung des Ausbildungsgesetzes (Ausbildungsgesetz) sowie die beigelegten weiteren Belehrungen sind bestehend dieses Vertrages

**Angaben zum Ausbildenden**

Apotheke am Markt  
Name des Ausbildenden (Ausbildungsbetrieb):  
Marktstraße 1  
Straße, Haus-Nr.:  
PLZ Ort  
E-Mail-Adresse (Angabe freiwillig)  
0800 123482394  
Telefonnummer  
Beispiel, Petta  
Name, Vorname verantwortliche Ausbilderin  
Angaben zum/zu gesetzlichen Vertreter(n)  
keiner Eltern Mutter Vater Vormund  
Name, Vorname  
Straße, Haus-Nr.:  
PLZ Ort  
Name, Vorname  
Straße, Haus-Nr.:  
PLZ Ort

**Angaben zum Auszubildenden**

Mustermann Max  
Name  
Musterrstraße 4  
Straße, Haus-Nr.:  
PLZ Ort  
01.03.2002 Geburtsdatum  
E-Mail-Adresse (Angabe freiwillig)  
0800 23834570  
Mobile/Telefonnummer (Angabe freiwillig)

**§ 1 – Dauer der Ausbildung**

1. Dauer  
Die Ausbildungsdauer beträgt nach der Ausbildungsvorordnung  
36 Monate.  
Auf die Ausbildungsdauer wird die Berufsausbildung zurück<sup>2</sup>  
bzw. eine berufliche Fortbildung in  
mit Monaten angeholt.<sup>3</sup>

Die Berufsausbildung wird in  
keiner Eltern Mutter Vater Vormund  
Vollzeit Teiltzeit (%) der Ausbildungsdauer in Vollzeit durchgeführt.  
Die Ausbildungsdauer verlängert sich aufgrund der Teilzeit um Monate.  
Die Ausbildungsdauer verkürzt sich vorbehaltlich der Entscheidung der zuständigen Stelle aufgrund  
um Monate.  
Soviel keine gesetzlich erlaubte Formulierung gewählt wird, steht dies allein der Vereinbarung der Leistungsfähigkeit. Auch darf werden alle Menschen angesprochen – unabhängig von ihrem Geschlecht (siehe).

**§ 2 – siehe S. 3 des Berufsausbildungsvertrages**

**§ 3 – Ausbildungsstätte**  
Die Ausbildung findet vorbehaltlich der Regelungen nach § 4 Nr. 12 dieses Vertrages in

Name/Ort der Ausbildungsstätte  
und den mit dem Betrieb für die Ausbildung (bitteweise zusammenhängende Bau-, Montage- und sonstigen Arbeitsstellen) statt

**§ 4 – Pflichten des Ausbildenden**  
Ausbildungsmethoden aufzuhalten der Ausbildungsstätte) und für den folgenden Zeitraum in den/deren (späteren) Ausbildungsstätte(n) vorgesehen (hierzu zählen auch Auslandsaufenthalte)

**§ 5 – Pflichten des/dieses Auszubildenden**  
Führung von schriftlichen oder elektronischen Ausbildungsnachweisen  
folgt geführt.  
Der Ausbildungsnachweis wird

**§ 6 – Bestandteile der Vergütung und sonstige Leistungen**  
Höhe und Fälligkeit  
Das Ausbildungsvorhaben fällt in den Geltungsbereich des folgenden Tarifvertrages:  
Das Ausbildungsvorhaben fällt nicht in den Geltungsbereich eines gültigen Tarifvertrages.

**§ 13 – Vertragstafel**  
Der Auszubildende verzögert sich über die Ausbildungsdauer und deren gewährte Vermehrung und Vertreterin die Vertragsabfassung verzögert noch dem Erreichung ausreichend. Bei elektronischer Abfassung ist die Vertragsabfassung so zu überprüfen, dass die elektronische Unterschrift der beteiligten Parteien eindeutig ist. Deine Auszubildende verzögert sich, den Empfang der elektronischen Vertragsabfassung selbst oder einer anderen Person, die die elektronische Unterschrift der beteiligten Parteien übermittelt. Die Vertragsabfassung und der Empfangsbeleg sind von deinen Auszubildenden nach Abschluss der Ausbildungsvorhaben zu überprüfen. Der Vertrag ist abgeschlossen, wenn die beteiligten Parteien unterschrieben haben.

Vorliegende Vertrag ist  
gem. § 13 Satz 1 zweifel (bei Mündlich –fach) ausgestellt  
und von Vertragsabschließenden ehrlichst unterschrieben worden.  
gemäß § 13 Satz 2 elektronisch abgesetzt und übermittelt worden.

Anlage gemäß § 4 Nr. 1 des Berufsausbildungsvertrages<sup>4</sup>  
**date\_document**

Musterrat, 20.09.2022  
Ort, Datum  
Max Mustermann  
Unterschrift des Auszubildenden  
Mustermann  
Unterschriften des gesetzlichen Vertreters  
stamp\_company signature\_company signature\_legal\_guardian

(a) Erste Seite

(b) Zweite Seite

Abbildung 2.2: Ausbildungsvertrag der Industrie- und Handelskammer

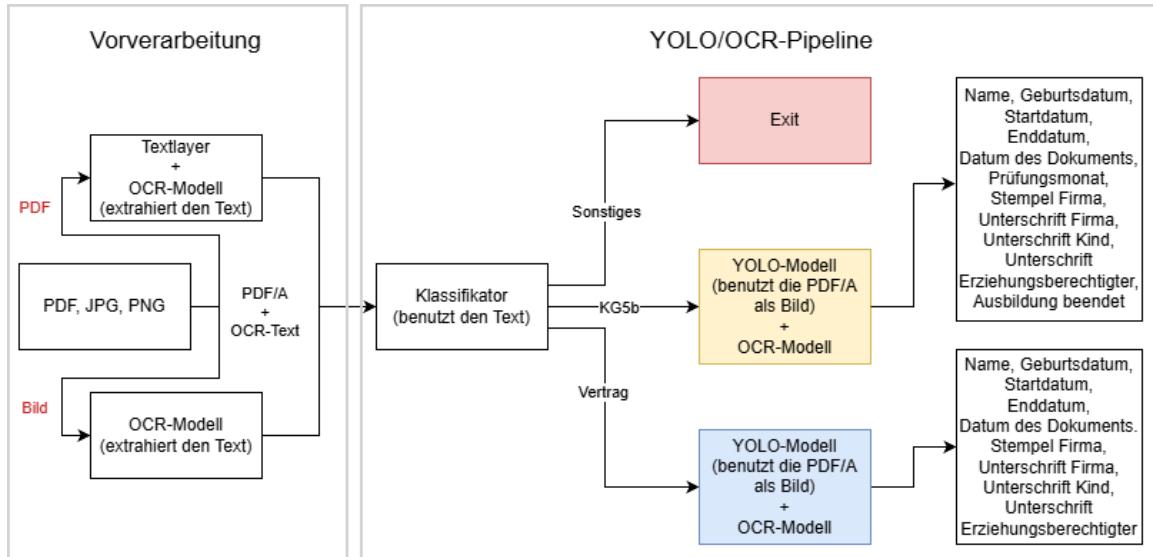


Abbildung 2.3: Schematischer Workflow der aktuellen AuBe-Pipeline

der Text extrahiert, während bei einer Bilddatei lediglich das OCR-Modell zum Einsatz kommt. Als Ergebnis der ersten Stufe werden dann ein PDF/A und der extrahierte Text zurückgeliefert.

Auf Basis des extrahierten Textes wird in der nächsten Stufe zu Beginn eine Klassifikation durchgeführt. Hierbei unterscheidet der Klassifikator zwischen den bereits vorgestellten Dokumententypen: KG5b, Vertrag und Sonstiges. Wurde das Dokument als **Sonstiges** klassifiziert, endet an dieser Stelle die Bearbeitung. Handelt es sich hingegen um ein KG5b oder einen Vertrag, wird je nach Dokumententyp eine Erkennung mit einem YOLO-Modell gestartet. Innerhalb der Bounding-Boxen wird der Text extrahiert und dem jeweiligen Label zugeordnet.

Schlussendlich stehen die erkannten Informationen zur weiteren Verarbeitung bereit.

## 2.2.2 Rahmenbedingungen und Infrastruktur

Die Entwicklung und Evaluation der Modelle erfolgt unter datenschutzrechtlichen Auflagen. Da im Rahmen dieser Arbeit personenbezogene Echtdaten verarbeitet werden, wird eine isolierte On-Premises-Infrastruktur verwendet.

Die technische Architektur ist in Abbildung 2.4 schematisch dargestellt.

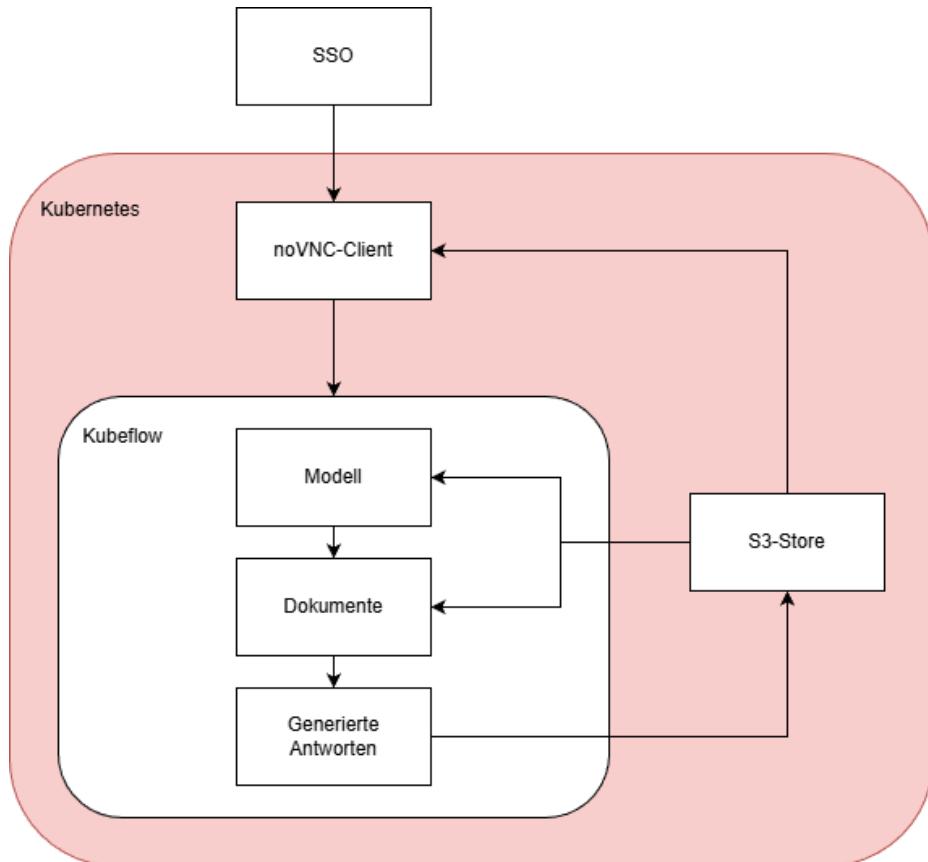


Abbildung 2.4: Schematische Darstellung der Trainings- und Inferenzinfrastruktur

Das System basiert auf einem abgeschotteten Kubernetes-Cluster. Für die rechenintensiven Aufgaben, insbesondere das Fine-Tuning und die Inferenz der VLMs, stehen innerhalb des Clusters zwei NVIDIA A40 GPUs mit jeweils 48 GB VRAM[NVID 22] zur Verfügung. Um die Bereitstellung der VLMs zu erleichtern, wird auf dem Cluster Kubeflow verwendet. Kubeflow ist eine Open-Source-Plattform, die speziell für das Entwickeln, Trainieren und Deployen von Machine-Learning-Modellen konzipiert wurde.

Der Zugang zum Cluster sowie zur Integrated Development Environment (IDE) erfolgt über eine noVNC-Schnittstelle (browserbasierter Remote-Desktop).

Um einen Test- oder Trainingslauf durchzuführen, wird der folgende Prozess durchlaufen:

1. **Initialisierung:** Ein Python-Skript in der IDE startet den Kubeflow-Job.
2. **Datenbereitstellung:** Die Pipeline lädt sowohl das VLM als auch die zu klassifizierenden Dokumentenbilder aus dem S3-Store in den GPU-Cluster.
3. **Verarbeitung:** Die Inferenz oder das Training findet im Kubernetes-Cluster statt.
4. **Persistierung:** Die Ergebnisse werden zurück in den S3-Store geladen.

## 2.3 Vision Language Models (VLMs)

Ein VLM besteht aus drei Komponenten: einem Image-Encoder, einem Adapter und einem Large Language Model (LLM). In Abbildung 2.5 wird der Aufbau dargestellt.



Abbildung 2.5: Architektur eines Vision Language Models

Der Image-Encoder verarbeitet die Bildeingabe und extrahiert visuelle Merkmale (Features). Zu diesem Zweck kommen häufig vortrainierte Modelle wie Vision Transformer (ViT) oder CLIP zum Einsatz. Diese Modelle zerlegen ein Bild in kleinere Patches, die ähnlich wie Token in Sprachmodellen behandelt werden. Jeder Patch wird in einen Vektor überführt, der die Eigenschaften dieses Bildausschnitts repräsentiert. Das Ergebnis ist eine Sequenz von Bildvektoren, die die visuellen Informationen des gesamten Bildes beinhaltet.

Der Adapter verbindet den Image-Encoder mit dem Sprachmodell. Diese Schicht transformiert die Ausgabe des Image-Encoders in ein Format, das mit den Textvektoren des Sprachmodells kompatibel ist. In vielen Modellen besteht der Adapter aus einer linearen Schicht oder einem kleinen neuronalen Netz. Die Bildvektoren des Image-Encoders werden in denselben Vektorraum wie die Textvektoren des LLM projiziert.

Die Hauptkomponente bildet das LLM, welches für die eigentliche Verarbeitung und Generierung zuständig ist. Das LLM erhält die Textvektoren zusammen mit den Bildvektoren und fusioniert diese. Mithilfe des Attention-Mechanismus werden Beziehungen zwischen visuellen und textuellen Elementen verstanden. So ist es möglich, Fragen zu Bildinhalten zu beantworten oder Bildbeschreibungen zu generieren.

Die größte Herausforderung dieser modernen Technologie ist der hohe VRAM-Verbrauch. Neben der statischen VRAM-Nutzung kommt mit jeder Anfrage ein dynamischer Verbrauch dazu. Dabei setzt sich der statische Verbrauch aus den Gewichten der Modelle zusammen, während sich der dynamische Verbrauch aus den Key-Value-Caches und den Aktivierungen bildet.

## 2.4 Vorstellung der Modelle

Für die Klassifikation und die Extraktion der Informationen aus den Dokumenten wurden drei verschiedene VLMs evaluiert.

### 2.4.1 Pixtral-12B

Das Pixtral-12B-2409 [[Pixt 24](#)] wurde im Jahr 2024 von Mistral AI veröffentlicht. Es basiert auf einem 12 Milliarden Parameter großen Text-Decoder mit einem zusätzlichen 400 Millionen Parameter umfassenden Vision-Encoder. Es wurde speziell auf das Verständnis von Bildern und Dokumenten trainiert, weshalb es einen optimalen Kandidaten für die vorliegende Arbeit darstellt. Mit einem theoretischen Kontextfenster von 128.000 Token ermöglicht das Modell die gleichzeitige Verarbeitung mehrerer Bilder [[Agra 24](#)].

### 2.4.2 Qwen-2.5-VL

Das Qwen2.5-VL-7B-Instruct[[Qwen 25b](#)] sowie das Qwen2.5-VL-32B-Instruct[[Qwen 25a](#)] wurden im Jahr 2025 von Alibaba Cloud veröffentlicht. Die beiden Modelle basieren auf dem gleichen Image-Encoder mit 600 Millionen Parametern. Lediglich die Größe des Text-Decoders ist mit 7 Milliarden beziehungsweise 32 Milliarden Parametern unterschiedlich. Eine Besonderheit dieser Modelle ist, dass sie speziell auf das Verarbeiten von Dokumenten trainiert wurden[[Bai 24](#)].

### 2.4.3 Benchmarks

Im Folgenden sind die Benchmarks der einzelnen Modelle gelistet. Besonders das Ergebnis des DocVQA-Benchmarks ist von Interesse, da hier ein Visual Question Answering (VQA) auf Dokumentenbildern durchgeführt wird[[Mine 21](#)].

Modell	<b>Pixtral-12B</b>	<b>Qwen2.5-VL-7B</b>	<b>Qwen2.5-VL-32B</b>
<b>DocVQA</b>	90,7	95,7	94,8
<b>MMMU</b>	52,0	58,6	70,0

Tabelle 2.1: Benchmark-Ergebnisse der evaluierten Modelle[[Bai 24](#), [Qwen 25a](#), [Agra 24](#)]

## 2.5 Parameter-Efficient Fine-Tuning

Das Anpassen der Gewichte eines bereits trainierten Modells an eine spezifische Domäne setzt eine Infrastruktur mit hoher Rechenleistung voraus. Mithilfe von Parameter-Efficient Fine-Tuning (PEFT) kann die Anzahl der zu trainierenden oder neuen Parameter signifikant gesenkt werden. Infolgedessen verringert sich auch der Rechenaufwand für das Fine-Tuning der Modelle erheblich[[Mang 22](#)].

Eine der bekanntesten Methoden im Bereich PEFT ist die Low-Rank Adaptation (LoRA). Anstatt das gesamte Modell neu zu trainieren, friert LoRA die ursprünglichen Gewichte ein und fügt stattdessen trainierbare Matrizen mit einem niedrigen Rang  $r$  in jede Schicht der Transformer-Architektur ein. Um Rechenaufwand und Speicherplatz zu sparen, werden ausschließlich diese kleineren Matrizen trainiert[[Hu 21](#)].

Ergänzend zu LoRA wird die Variante Rank-Stabilized LoRA (rsLoRA) betrachtet. Bei der Standard-LoRA führt ein steigender Rang  $r$  oft nicht zu einer besseren Performance, da der verwendete Faktor ( $\alpha/r$ ) das Lernen bei höheren Rängen verlangsamen oder hemmen kann. rsLoRA stabilisiert diesen Prozess, indem die Adapter durch die Quadratwurzel des Rangs ( $\alpha/\sqrt{r}$ ) geteilt werden[[Kala 23](#)].

## 2.6 Verwandte Arbeiten

Ein Feld gilt hierbei als True Positive (TP), wenn sowohl der Key des Feldes vorhanden ist als auch der Value des Feldes mit dem der Ground Truth übereinstimmt[Huan 21].

## Kapitel 3

### Methodik und Implementierung

Um die Eignung der VLMs als potenziellen Ersatz für die YOLO/OCR-Pipeline zu beurteilen, werden die folgenden Forschungsfragen beantwortet:

1. Wie verhält sich ein nicht domänen spezifisch angepasstes VLM hinsichtlich Latenz, Klassifikations- und Extraktionsleistung im Vergleich zur YOLO/OCR-Pipeline, und welche Vorteile bietet ein einzelnes generalistisches Modell gegenüber spezialisierten Einzelsystemen?
2. Welche Auswirkungen hat ein domänen spezifisches Training eines kleineren Modells auf dessen Latenz, Klassifikations- und Extraktionsleistung im Vergleich zu einem leistungs stärkeren Basismodell?
3. Wie unterscheidet sich der Ressourcenverbrauch (Energieverbrauch und VRAM-Nutzung) der VLMs von dem der YOLO/OCR-Pipeline?

Abbildung 3.1 zeigt einen Überblick der gesamten Methodik.

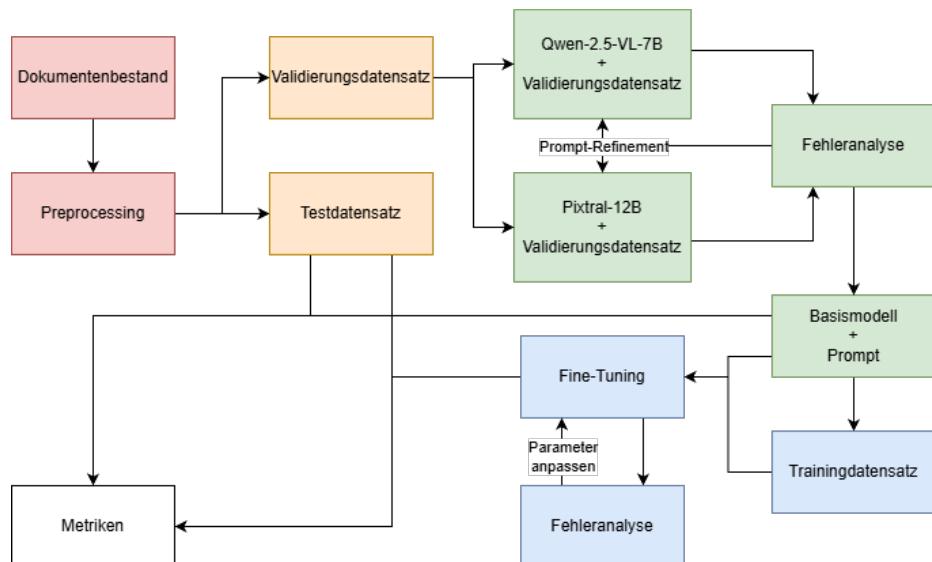


Abbildung 3.1: Überblick der Methodik

## 3.1 Modellierung des Dokumentenbestands als Grundlage der Evaluation

### 3.1.1 Preprocessing des Dokumentenbestands

Durch Datenlieferungen der Familienkasse stehen für die Evaluation mehrere Tausend Dokumente als PDF oder Bilddatei zur Verfügung. Um einen einheitlichen Dokumentenkorpus zu schaffen und die Klassifikations- und Extraktionsleistung zu verbessern, erfolgt eine Vorverarbeitung der Dokumente. Dieser Prozess ist in Abbildung 3.1 in Rot dargestellt.

Im Fokus des Preprocessings stehen die Korrektur der Bildausrichtung sowie die Konvertierung der PDF-Dokumente in Bilddateien. Zur Korrektur der Rotation werden die Exchangeable Image File Format (EXIF)-Metadaten ausgelesen, um die ursprüngliche Ausrichtung wiederherzustellen. Während ältere Modelle Bilder auf eine feste Dimension skalierten, unterstützen die in dieser Arbeit evaluierten Modelle eine dynamische Skalierung. Zur Begrenzung der maximalen Token-Anzahl werden die Bilder dennoch auf eine maximale Pixelanzahl skaliert, wobei das ursprüngliche Seitenverhältnis beibehalten wird. Listing 1 zeigt die Korrektur der Bildausrichtung und Skalierung innerhalb des Preprocessings.

---

```

from pathlib import Path
from PIL import Image, ImageOps
from pdf2image import convert_from_path

def preprocessing(path_to_document: str, max_pixels: int) -> [Image]:
    ...
    for _, page in enumerate(pages):
        # corrects the image orientation
        page = ImageOps.exif_transpose(page)

        if page.width * page.height > max_pixels:
            scale_factor = (max_pixels / (page.width * page.height)) ** 0.5

            new_width = int(page.width * scale_factor)
            new_height = int(page.height * scale_factor)

            page_resized = page.resize((new_width, new_height), Image.Resampling.LANCZOS)
    ...

```

---

Listing 1: Preprocessing der Dokumente

### 3.1.2 Modellierung der Dokumentenarten als JSON-Objekte

Für die standardisierte Weiterverarbeitung der Modellausgaben ist eine konsistente Struktur essenziell. Da moderne VLMs darauf trainiert sind, Antworten im JavaScript Object Notation (JSON)-Format zu liefern, werden die extrahierten Informationen in ein vordefiniertes JSON-Schema überführt.

Ein wesentlicher Vorteil gegenüber der herkömmlichen Pipeline besteht in der simultanen Klassifikation sowie der IE in einem einzigen Inferenzschritt. Die JSON-Schemata für die relevanten Dokumentenarten sind in den Listings 2 und 3 dargestellt. Bei dem JSON-Schema der sonstigen Dokumente in Listing 4 findet nur die Klassifikation statt, da hier keine Informationen relevant sind. Die Felder der JSON-Schemata bilden die relevanten Informationen aus Abschnitt 2.1 ab.

---

```
{
  "file_name": {
    "type": "kg5b",
    "name_child": "name, vorname",
    "birthday_child": "DD.MM.YYYY",
    "start_date_apprenticeship": "DD.MM.YYYY",
    "end_date_apprenticeship": "DD.MM.YYYY",
    "date_document": "DD.MM.YYYY",
    "stamp_company": true,
    "signature_company": true,
    "signature_child": true,
    "signature_legal_guardian": true,
    "apprenticeship_finished": true,
    "exam_month": "MM"
  }
}
```

---

Listing 2: JSON-Schema des Dokumententyps KG5b

---

```
{
  "file_name": {
    "type": "vertrag",
    "name_child": "name, vorname",
    "birthday_child": "DD.MM.YYYY",
    "start_date_apprenticeship": "DD.MM.YYYY",
    "end_date_apprenticeship": "DD.MM.YYYY",
    "date_document": "DD.MM.YYYY",
    "stamp_company": true,
    "signature_company": true,
    "signature_child": true,
    "signature_legal_guardian": true
  }
}
```

---

Listing 3: JSON-Schema des Dokumententyps Ausbildungsvertrag

---

```
{
  "file_name": {
    "type": "sonstiges"
  }
}
```

---

Listing 4: JSON-Schema des Dokumententyps Sonstiges

Die Klassifikation wird im JSON-Objekt über das Feld `type` abgebildet und ist in allen JSON-Schemata vorhanden. Felder der IE sind über die Dokumentenarten nicht konsistent, jedoch bildet das Schema eines Vertrags eine Teilmenge des Schemas eines KG5b-Formulars. Das Schema des KG5b-Formulars beinhaltet zusätzlich die Felder `apprenticeship_finished` und `exam_month`.

Bei den binären, booleschen Feldern markiert ein `true` das Vorhandensein eines Merkmals, während ein `false` dessen Fehlen oder das Nicht-Erkennen repräsentiert. Um eine strukturelle Konsistenz zu erzwingen, werden Felder, bei denen die dazugehörige Information im Dokument fehlt, bei booleschen Typen mit `false` und bei Textfeldern mit einer leeren Zeichenkette belegt.

Da die Dokumente in den meisten Fällen aus mehreren Seiten bestehen, repräsentiert ein JSON-Objekt das mehrseitige Dokument. Sollten Dokumente verschiedener Arten vermischt vorliegen, generiert das Modell für jede erkannte Dokumentenart ein separates JSON-Objekt.

### 3.1.3 Test- und Validierungsdatensatz

Die Erstellung des Test- sowie des Validierungsdatensatzes ist im methodischen Überblick (Abbildung 3.1) orange hervorgehoben und schließt unmittelbar an die Vorverarbeitung der Dokumente an. Beide Datensätze umfassen jeweils 60 Dokumente, aufgeteilt in 20 Beispiele pro Dokumentenart. Um die reale Datenverteilung bestmöglich abzubilden, werden die Dokumente manuell ausgewählt.

Die Datensätze decken dabei verschiedene Herausforderungen ab. Neben Dokumenten mit geringer Qualität, vorrangig **\*\*mangelhaften Scans und Fotos\*\***, befinden sich fehlerhaft ausgefüllte Dokumente sowie Dokumente mit fehlenden Unterschriften und Stempeln in den Datensätzen. Um die hohe Varianz der Verträge und sonstigen Dokumente abzubilden, enthalten die Datensätze viele unterschiedliche Layouts. Die Exemplare für die Verträge beinhalten sowohl starre Layouts als auch Fließtexte unterschiedlicher Firmen und Kammern. Die sonstigen Dokumente beinhalten keine kontextfremden Dokumente, sondern Schulbescheinigungen, Studienbescheinigungen und Eintragungen bei der Handwerkskammer, um die Komplexität der Unterscheidung zu erhöhen.

Zur Erstellung der Wahrheitswerte (Ground Truth) werden alle Dokumente manuell annotiert und in das entsprechende JSON-Schema überführt.

## 3.2 Evaluation der generierten JSON-Objekte

### 3.2.1 Validierung des VLM-Outputs

Um eine korrekte Formatierung des JSON-Outputs der VLMs sicherzustellen, existieren verschiedene Ansätze. Während komplexere Lösungen nur Tokens zulassen, die syntaktisch in JSON-Objekten enthalten sein könnten (Constrained Decoding), wird in dieser Evaluation ein direkterer Ansatz gewählt. Hierbei werden aus der generierten Antwort potenzielle JSON-Objekte extrahiert und geparsst. Schlägt das Parsen der JSON-Objekte fehl, werden diese zurück an das VLM geliefert, mit der Anweisung, diese zu korrigieren. Das Modell bekommt insgesamt drei Versuche, ein valides JSON-Objekt zu generieren, bevor das Dokument als ungültig eingestuft wird.

### 3.2.2 Vergleichslogik der unterschiedlichen JSON-Felder

Im Gegensatz zur Verarbeitung von Antworten herkömmlicher Modelle weisen VLMs eine hohe Varianz an Ausgabeformaten auf. Verschiedene Feldtypen der generierten JSON-Objekte müssen deshalb spezifisch normalisiert werden, um eine Grundlage für den Vergleich

zu bilden. Des Weiteren müssen Feldtypen wie Namen, Booleans, Zeichenketten, Datumsangaben und Monate nach unterschiedlichen Logiken verglichen werden. Für den Vergleich werden auf Basis der Klasse in Listing 5 verschiedene Komparatoren (Comparator) definiert.

---

```
from typing import Dict

class Comparator:
    def compare(self, pred_value, gt_value) -> Dict:
        pass
```

---

Listing 5: Basisklasse der verschiedenen Comparator

Der `NameComparator` in Listing 6 vergleicht die Namensfelder mithilfe der Levenshtein-Similarity, um kleinere Fehler und verschiedene Schreibweisen zu tolerieren. Beispielsweise wird ein Name, der `ue` statt `ü` enthält, nicht als Fehler gewertet. Der Threshold, der zwischen korrekt und nicht korrekt entscheidet, liegt bei 0,8. Da die YOLO/OCR-Pipeline denselben Vergleich mit identischem Threshold verwendet, wird eine Vergleichbarkeit der Ergebnisse gewährleistet.

---

```
from rapidfuzz import fuzz

class NameComparator(Comparator):
    def __init__(self, threshold=0.80):
        self.threshold = threshold

    def compare(self, pred_value: str, gt_value: str) -> Dict:
        # calculate levenshtein-similarity
        similarity = fuzz.ratio(pred_value.lower(), gt_value.lower()) / 100.0
        is_correct = similarity >= self.threshold

    ...
```

---

Listing 6: Comparator für Namen

Der `BooleanComparator` normalisiert boolesche Felder entsprechend der Python-Syntax zu `True` und `False`. Zu den akzeptierten Varianten gehören sowohl `true` und `false`, 0 und 1, `wahr` und `falsch` als auch `vorhanden` und `nicht vorhanden`.

Zeichenketten, die keine Namen darstellen, werden durch den `ExactComparator` verglichen. Hierbei findet keine Normalisierung statt, sodass die Zeichenketten exakt übereinstimmen müssen.

Datumsangaben werden durch den `DateComparator` einheitlich in das Format DD.MM.YYYY überführt. Obwohl dies nicht dem internationalen Standard entspricht, erweist sich dieses Vorgehen als am robustesten, da die vorliegenden deutschen Dokumente primär in diesem Format ausgefüllt sind.

Der `MonthComparator` normalisiert alphanumerische Monate in eine numerische Darstellung. Falls vollständige Datumsangaben geliefert werden, wird der `DateComparator` genutzt, um den Monat zu extrahieren.

Um die verschiedenen Comparator den Feldern zuzuordnen, existiert für jede Dokumentenart eine Konfiguration, die dem jeweiligen JSON-Schema aus Abschnitt 3.1.2 entspricht. Das Schema 7 definiert beispielhaft die Konfiguration für das KG5b-Formular.

---

```
import ExactComparator, NameComparator, DateComparator, BooleanComparator, MonthComparator

CONFIG_KG5B = {
    "type": {"type": "string", "comparator": ExactComparator()},
    "name_child": {"type": "string", "comparator": NameComparator()},
    "birthday_child": {"type": "date", "comparator": DateComparator()},
    "start_date_apprenticeship": {"type": "date", "comparator": DateComparator()},
    "end_date_apprenticeship": {"type": "date", "comparator": DateComparator()},
    "date_document": {"type": "date", "comparator": DateComparator()},
    "stamp_company": {"type": "boolean", "comparator": BooleanComparator()},
    "signature_company": {"type": "boolean", "comparator": BooleanComparator()},
    "signature_child": {"type": "boolean", "comparator": BooleanComparator()},
    "signature_legal_guardian": {"type": "boolean", "comparator": BooleanComparator()},
    "apprenticeship_finished": {"type": "boolean", "comparator": BooleanComparator()},
    "exam_month": {"type": "date", "comparator": MonthComparator()}
}
```

---

Listing 7: Konfiguration des KG5b-Formulars

### 3.2.3 Bewertung der Klassifikation und der Information Extraction

Um die generierten JSON-Objekte der VLMs schlussendlich bewerten zu können, werden die Klassifikation und IE unabhängig voneinander bewertet. Für die Evaluation werden die einzelnen JSON-Felder jedes Dokuments mithilfe der in der Konfiguration definierten Comparator mit der Ground Truth verglichen und verschiedene Metadaten erfasst. Die Metadaten beinhalten den Status (vorhanden, nicht vorhanden oder halluziniert), die Korrektheit sowie die Fehlerart des Feldes. Halluzinierte Felder sind definiert als Felder, die nicht in der jeweiligen Konfiguration vorkommen. Das Ergebnis ist ein Python-Dictionary (Listing 8), das für jedes JSON-Objekt eines Dokuments die Metadaten für jedes Feld beinhaltet.

---

```

result = {
    "document1": {
        "type": {
            "status": "present",
            "is_correct": True,
            "error": None
        },
        "name_child": {
            "status": "missing",
            "is_correct": None,
            "error": None
        },
        "birthday_child": {
            "status": "present",
            "is_correct": False,
            "error": "format_error"
        }
    ...
}
...
}

```

---

Listing 8: Ergebnis-Dictionary mit den Metadaten des gesamten Dokumentenkorpus

Auf Basis dieses Dictionaries wird für die Bewertung der Klassifikation aus jedem Dokument das Feld `type` extrahiert. Da das Feld `type` in jeder JSON der Dokumentenarten (siehe Listings ??) vorkommt, kann es nicht halluziniert werden. Jedoch kann das Feld fehlen, weshalb neben den Klassen KG5b, Vertrag und Sonstiges eine weitere Klasse `missing` eingeführt wird. `Missing` wird in der Auswertung nur betrachtet, wenn mindestens ein Dokument in der Klasse vorkommt. Für die Bestimmung der Güte der Modelle wird eine Confusion-Matrix erstellt, wobei der Fokus auf dem F1-Score liegt.

Die Bewertung der Information Extraction ist komplexer als die der Klassifikation. Um den Entity-Level F1-Score, der die Hauptmetrik für den Vergleich darstellt, zu berechnen, werden die Metadaten aller Felder außer dem Feld `type` benötigt.

Auf Basis dieser Metadaten werden für den gesamten Dokumentenkorpus die True Positives (TP), False Positives (FP) und False Negatives (FN) gezählt. Ein Feld zählt als TP, wenn es vorhanden sowie korrekt ist. Zu den FN gehören Felder, die entweder fehlen oder vorhanden, aber falsch sind. Ein Feld, das vorhanden, aber falsch ist, zählt zusätzlich zu den FP, zusammen mit den Feldern, die halluziniert wurden. Diese doppelte Bestrafung stellt eine sehr strenge Bewertung für inhaltliche Fehler dar. True Negatives (TN) sind nichtzählbar, da die Menge an nicht gefundenen, leeren Feldern theoretisch unendlich groß ist. Durch die Konfiguration und die Ground Truths ist eindeutig feststellbar, welche Felder halluziniert

sind. Da mithilfe der Konfiguration auch ohne Ground Truth die benötigten Felder gefiltert werden können, kann wie in Listing 9 entschieden werden, ob die Halluzinationen bestraft werden.

---

```
def calculate_field_based_f1_score(gt_jsons,
                                    pred_jsons,
                                    penalize_halluzinations: bool = False) -> float:
    ...
    fp = count_correct_fields

    if penalize_halluzinations:
        fp += count_halluzinated_fields
    ...

```

---

Listing 9: Parameter, der bestimmt, ob Halluzinationen bestraft werden

### 3.3 Messung von Latenz, Energieverbrauch und VRAM-Nutzung

Im Hinblick auf die Bewertung der Modelle werden neben der Güte der Klassifikation und IE die Latenz, die VRAM-Nutzung sowie der Energieverbrauch gemessen.

Die Latenz der Modelle stellt lediglich das Delta zwischen Start- und Endzeit der Inferenz dar. Hierbei werden die Zeitpunkte gemessen, an denen die eigentliche Inferenz startet und die Antwort bereitsteht. Das Laden des Modells wird nicht betrachtet, da dieses keinen Einfluss auf die Antwortzeit in der produktiven Umgebung hat und folglich nicht relevant für die Evaluation ist.

Des Weiteren wird der Energieverbrauch während der Inferenz in Joule erfasst. Die Messung der Leistung in Watt ist nicht zielführend, da sie die Zeit nicht berücksichtigt und Modelle mit geringerer Latenz bei vergleichbarem Energieverbrauch benachteiligt werden würden.

Ebenso wie der Energieverbrauch wird der dynamische VRAM-Verbrauch während der Inferenz gemessen. Mit der Initialisierung des Modells wird zudem einmalig der statische Verbrauch gemessen.

### 3.4 Auswahl des Basismodells

Die Auswahl eines geeigneten Basismodells ist ein wichtiger Schritt für das spätere Fine-Tuning. Wie in Abbildung 3.1 (grün markierte Felder) dargestellt, erfolgt diese Auswahl in einem iterativen Prozess. Ziel ist es, das Modell zu identifizieren, das bereits ohne Training die beste Leistung sowohl bei der Klassifikation als auch bei der IE erreichen kann.

In jeder Iteration werden die Modelle Pixtral-12B und Qwen-2.5-VL-7B mit dem Validierungsdatensatz 3.1.3 evaluiert. Die Iteration gliedert sich in folgende Schritte:

1. Evaluation: Durchführung der Inferenz beider Modelle mit dem Validierungsdatensatz.
2. Fehleranalyse: Untersuchung der Fehler in der Klassifikation und in der IE.
3. Prompt-Refinement: Anpassung des Prompts auf Basis der Fehleranalyse.

Der Prozess wird so lange durchlaufen, bis durch Änderungen des Prompts keine Steigerungen der Metriken mehr erzielt werden. Im Anschluss werden mit dem finalen Prompt und dem Testdatensatz 3.1.3 die finalen Metriken bestimmt und auf Basis dieser das Modell gewählt. Die Trennung von Validierungs- und Trainingsdatensatz schließt ein Data Leakage aus und verhindert ein Overfitting auf den Testdaten.

Bezogen auf das Prompt-Engineering wurde bewusst ein Zero-Shot-Ansatz gewählt. Im Gegensatz zu One-Shot- und Few-Shot-Prompting werden hier weniger Tokens verbraucht, was die Kosten pro Inferenz senkt. Des Weiteren kann das Modell so schneller antworten, da es weniger Input-Tokens verarbeiten muss.

Die Gestaltung des finalen Prompts (siehe Listing 10) folgt der R-K-F-Formel. Während der Kontext die vorliegenden Dokumentarten beschreibt und die Schemata definiert, gibt das Format die Ausgabeformatierung vor. Zum Einsparen von Tokens wird das Modell strikt angewiesen, lediglich das JSON-Objekt zu generieren.

Der hier evaluierte Prompt wird im Laufe der Arbeit auch für die Bewertung des Qwen-2.5-VL-32B genutzt.

---

```

prompt = """
    Du bist ein Dokumenten-Assistent.

    Analysiere die Dokumente und extrahiere die Daten im JSON-Format.
    Gib ausschließlich valides JSON zurück.
    Es werden folgende Dokumentarten unterschieden:

        1. KG5b: Ein offizielles Dokument der Bundesagentur für Arbeit zur Bescheinigung des
               Ausbildungsstatus. Wichtig: Der 'exam_month' entspricht nicht zwangsläufig dem
               'end_date_apprenticeship'. Das 'date_document' ist das Datum, an dem der
               Ausbildungsbetrieb unterschrieben hat.
        2. Vertrag: Ein klassischer Ausbildungsvertrag.
        3. Sonstiges: Alle Dokumente, die weder ein KG5b noch ein Vertrag sind.

    Schemas für die Dokumentarten:

    ...

    Extrahiere die Informationen aus den Bildern und wähle das passende Schema für die
    jeweilige Dokumentenart aus.
    Sollten mehrere Dokumentenarten vorliegen, erstelle für jede Art ein separates
    JSON-Objekt mit dem entsprechenden Schema.
    Fülle ausschließlich die im Schema definierten Felder aus.
    Gib keine weiteren Texte aus, sondern nur das valide JSON.

"""

```

---

Listing 10: Finaler Prompt der Modellauswahl

## 3.5 Fine-Tuning des Basismodells

### 3.5.1 Trainingsdatensatz

Um ein qualitativ hochwertiges Fine-Tuning zu ermöglichen, ist ein deutlich umfangreicherer Datensatz als für die Modellauswahl erforderlich. Der finale Trainingsdatensatz umfasst 610 Dokumente, die sich aus 227 Verträgen, 165 KG5b-Formularen und 218 sonstigen Dokumenten zusammensetzen.

Die Klassen sind bewusst ungleich verteilt. KG5b-Formulare haben ein starres Layout, wodurch schon mit einer geringeren Anzahl an Trainingsdaten ein gutes Ergebnis erwartet wird. Infolge der hohen Varianz der Verträge und der sonstigen Dokumente erhalten diese im Training eine höhere Gewichtung. Anders als beim Testdatensatz werden hier die Exemplare nicht manuell, sondern mithilfe einer Stichprobe ausgewählt. Dabei werden die Daten aus einem einwöchigen Zeitraum betrachtet, wodurch ohne manuelle Auswahl eine

genaue Repräsentation der realen Daten entsteht. Des Weiteren wird so ein Selection Bias verhindert.

Zur effizienten Erstellung der Ground Truth wird ein Model-Assisted Labeling eingesetzt. Hierbei generiert das im Prozess der Modellauswahl 3.4 gewonnene Modell, zusammen mit dem angepassten Prompt, für jedes Dokument das jeweilige JSON-Objekt. Im Anschluss werden diese JSON-Objekte manuell kontrolliert und korrigiert. Durch den Einsatz des Modells zur Ermittlung der Ground Truth konnte der Aufwand erheblich gesenkt werden.

Für das Fine-Tuning muss der Datensatz in das OpenAI Message Format (Listing 11) gebracht werden. Des Weiteren müssen die JSON-Objekte in Markdown-Blöcke eingebettet werden, um dem bereits gelernten Format des VLM zu entsprechen.

---

```

import json

response = f"```json {json.dumps(ground_truth)}```"

messages = [{"messages": [{"role": "user",
    "content": [{"type": "text", "text": prompt},
                {"type": "image", "image": image}],
    ...
  ],
  ...
  },
  {"role": "assistant",
    "content": [{"type": "text", "text": response}]}
]
...
]

```

---

Listing 11: OpenAI Message Format

### 3.5.2 Optimierung des Fine-Tunings

Das in Abschnitt 3.4 gewählte Basismodell wird mittels PEFT an die Dokumentenarten angepasst. Dabei kommen LoRA und die stabilisierte Variante rsLoRA zum Einsatz. Für das Training wird auf das Framework Unislot gesetzt, um den Ressourcenverbrauch gering zu halten. Des Weiteren wird das Modell in 4-Bit-Quantisierung geladen, um die VRAM-Auslastung weiter zu senken.

Der Systemprompt in Listing 12 ist pragmatisch gehalten, damit das Modell anhand von Beispielen statt durch Instruktion lernt.

---

```
prompt = """  
    Extract the relevant information from the documents and return it as JSON.  
"""
```

---

Listing 12: Prompt für das Fine-Tuning des Qwen-2.5-VL-7B

Der Trainingsdatensatz 3.5.1 wird mithilfe von Datenaugmentation an das Training angepasst. Vor jedem Trainingslauf wird der Datensatz vollständig randomisiert, um eine zufällige Reihenfolge der Trainingsdaten zu erhalten. Darüber hinaus wurden in einigen der Trainingsläufe die Seiten der Dokumente zufällig getauscht, um dem Modell anzutrainieren, Informationen unabhängig von der Position zu extrahieren.

Die Parameteroptimierung erfolgte iterativ, wie im blau markierten Bereich in Abbildung 3.1 zu erkennen ist, und es werden alle Komponenten des VLM trainiert. Die wichtigsten Parameter der finalen Konfiguration sind in Listing ?? abgebildet.

Um ein Overfitting zu vermeiden, wird neben dem Trainingsdatensatz auch hier der Validierungsdatensatz 3.1.3 verwendet. Nach 80 Dokumenten wird mit dem Validierungsdatensatz der Trainingsfortschritt gemessen. Am Ende des Trainingslaufs wird das Modell geladen, das den geringsten Loss gegenüber dem Validierungsdatensatz hatte.

---

```

from unsloth import FastVisionModel

model = FastVisionModel.get_peft_model(
    model,
    finetune_vision_layers=True,
    finetune_language_layers=True,
    finetune_attention_modules=True,
    finetune_mlp_modules=True,
    r=16,
    lora_alpha=16,
    lora_dropout=0,
    bias="none",
    random_state=3407,
    use_rslora=True,
    loftq_config=None,
)

```

---

```

from trl import SFTConfig, SFTTrainer

args = SFTConfig(
    eval_strategy="steps",
    eval_steps=10,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    neftune_noise_alpha=15,
    warmup_steps=15,
    num_train_epochs=2,
    learning_rate=5e-5,
    optim="adamw_8bit",
    lr_scheduler_type="cosine",
    max_seq_length=20000,
)

trainer = SFTTrainer(
    model = model,
    train_dataset = train_dataset,
    eval_dataset = eval_dataset,
    args = args,
    ...
)

```

---

Listing 14: Trainingsparameter des Fine-Tunings

Abschließend wird wie bei der Auswahl des Basismodells mit dem Testdatensatz 3.1.3 die finalen Metriken bestimmt.

# Kapitel 4

## Ergebnisse

### 4.1 Ergebnisse der YOLO/OCR-Pipeline gegenüber der Basismodelle

Die Evaluation der IE in Abbildung 4.1 zeigt deutliche Unterschiede zwischen den modernen VLMs und der YOLO/OCR-Pipeline.

Schwächen der YOLO/OCR-Pipeline zeigten sich bei der Verarbeitung von Ausbildungsverträgen. Hier erreichte die Pipeline einen F1-Score von 0,51. Im Vergleich schnitten insbesondere die beiden Qwen-Modelle deutlich besser ab. Das Qwen-2.5-VL-7B erkannte mit einem F1-Score von 0,83 und das Qwen-2.5-VL-32B mit einem F1-Score von 0,94 einen Großteil der Informationen. Das Pixtral-12B befindet sich mit einem F1-Score von 0,50 am unteren Ende der IE.

Bei den KG5b-Formularen liegt die Pipeline mit 0,75 nur knapp unter den Qwen-Modellen. Während das Qwen-2.5-VL-7B mit 0,8 und das Qwen-2.5-VL-32B mit 0,87 die stärksten Kontrahenten darstellen, erreichte auch hier das Pixtral-12B gerade einmal einen F1-Score von 0,46.

Die Ergebnisse der Klassifikation in Abbildung 4.2 verdeutlichen ein einheitlicheres Ergebnis. Besonders hervorzuheben ist das Qwen-2.5-VL-32B, das alle Dokumentarten am besten klassifiziert, während sich die restlichen Modelle mit einer maximalen Differenz von 0,03 in allen Dokumentarten auf einem Level befinden. Bei Betrachtung der in der Abbildung 4.3 beinhalteten Confusion Matrices fallen jedoch qualitative Unterschiede auf. Das Qwen-2.5-VL-32B klassifiziert die Verträge als auch KG5b-Formulare fehlerfrei. Im Gegensatz dazu erkennt die YOLO/OCR-Pipeline die sonstigen Dokumente perfekt, klassifiziert aber einige der Verträge und KG5b-Formulare als sonstige Dokumente. Das Pixtral-12B weist die größten Unsicherheiten auf und klassifizierte insgesamt 25 Dokumente als Vertrag. Im Vergleich dazu liefert das Qwen-2.5-VL-7B ein ähnliches Ergebnis, klassifizierte jedoch ein KG5b-Formular mehr richtig.

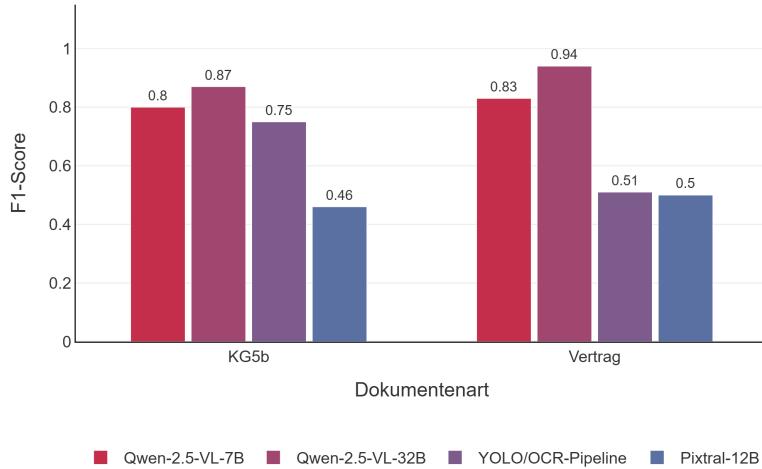


Abbildung 4.1: Ergebnisse der Information Extraction

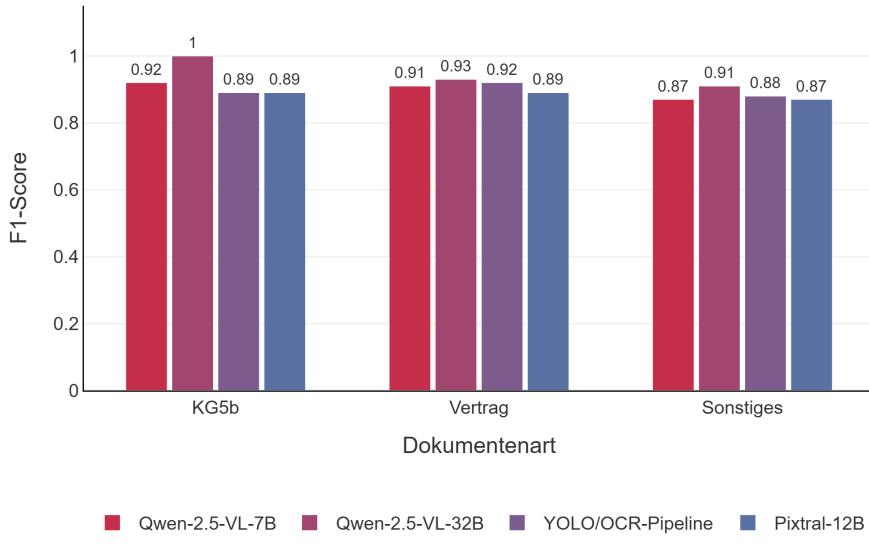


Abbildung 4.2: Ergebnisse der Klassifikation

Hinsichtlich der Latenz kehrt sich das Bild, wie in Abbildung 4.4 um. Die YOLO/OCR-Pipeline antwortet mit einer durchschnittlichen Latenz von 0,58 Sekunden nahezu sofort. Das kleinste VLM, das Qwen-2.5-VL-7B, benötigt mit durchschnittlich 7,43 Sekunden fast 13-mal länger als die Pipeline. Während das Pixtral mit durchschnittlich 10,28 Sekunden nur knapp über dem Qwen-2.5-VL-7B liegt, braucht das größte Modell im Durchschnitt 64,16 Sekunden

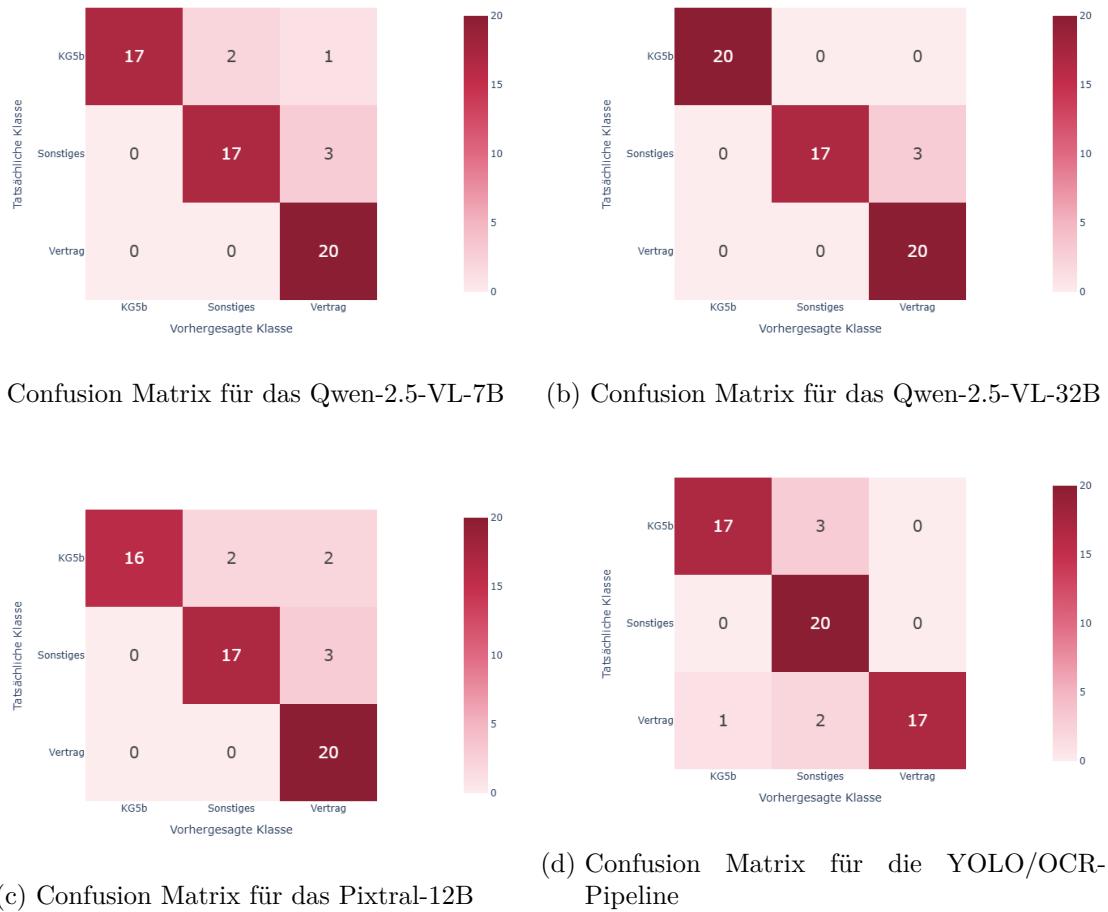


Abbildung 4.3: Confusion Matrices aller Modelle

## 4.2 Ergebnisse des weitertrainierten Modells gegenüber dem größeren Basismodell

Der Prozess der Auswahl des Basismodells ?? lieferte ein eindeutiges Ergebnis. Wie in Abbildung 4.1 und Abbildung 4.2 schlägt das Qwen-2.5-VL-7B das Pixtral-12B deutlich. Besonders in der IE stellt sich das Qwen-2.5-VL-7B als der bessere Kandidat für das Training heraus. Demnach wird im folgenden Abschnitt das trainierte Qwen-2.5-VL-7B (Qwen-2.5-VL-7B-finetuned) gegen das Qwen-2.5-VL-32B verglichen.

Trotz ähnlicher Ergebnisse zeigt das Qwen-2.5-VL-32B auch gegenüber dem angepassten Modell seine Stärken. Wie in Abbildung 4.5 zu erkennen, erreichten die beiden Modelle bei den KG5b-Formularen einen F1-Score von 0,87. Bei den Verträgen lag das Qwen-2.5-VL-32B mit einem F1-Score von 0,94 mit 0,10 über dem Qwen-2.5-VL-7B-finetuned.

Abbildung 4.6 zeigt eine geringe Differenz in der Klassifikationsleistung auf. Während beide Modelle die KG5b-Formulare perfekt zuordnen, liegt das Qwen-2.5-VL-7B-finetuned in

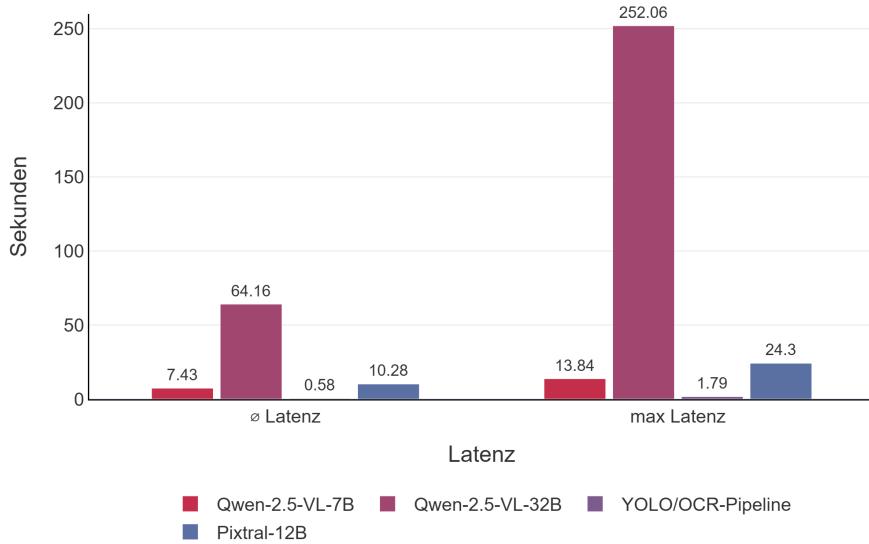


Abbildung 4.4: Ergebnisse der Latenzmessung

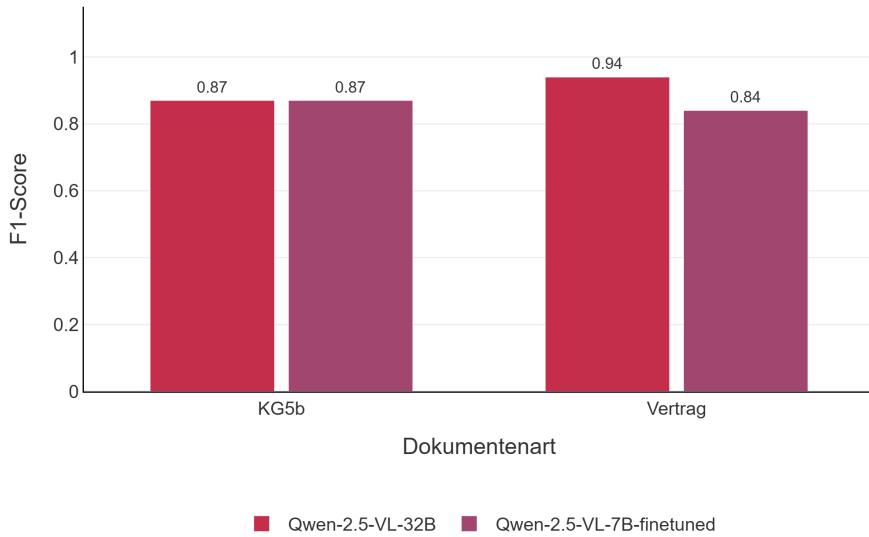


Abbildung 4.5: Ergebnisse der Information Extraction

den anderen Dokumentarten vorne. Hier erzielt das Modell einen F1-Score von 0,94 bei den Verträgen und 0,95 bei den sonstigen Dokumenten. Die Confusion Matrices in Abbildung 4.7 verdeutlichen dieses Ergebnis. Im direkten Vergleich klassifiziert das Qwen-2.5-VL-7B-finetuned einen Vertrag als sonstiges Dokument, während das Qwen-2.5-VL-32B drei sonstige Dokumente als Vertrag erkannte.

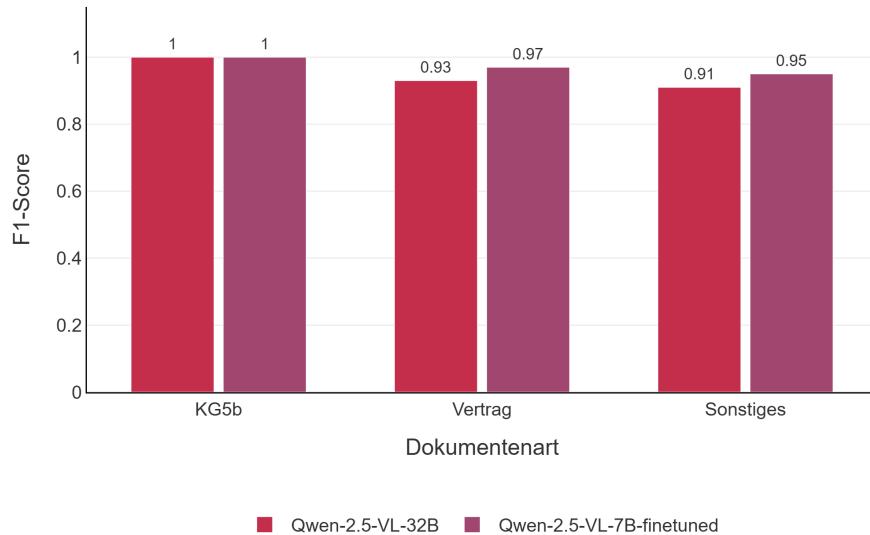


Abbildung 4.6: Ergebnisse der Klassifikation

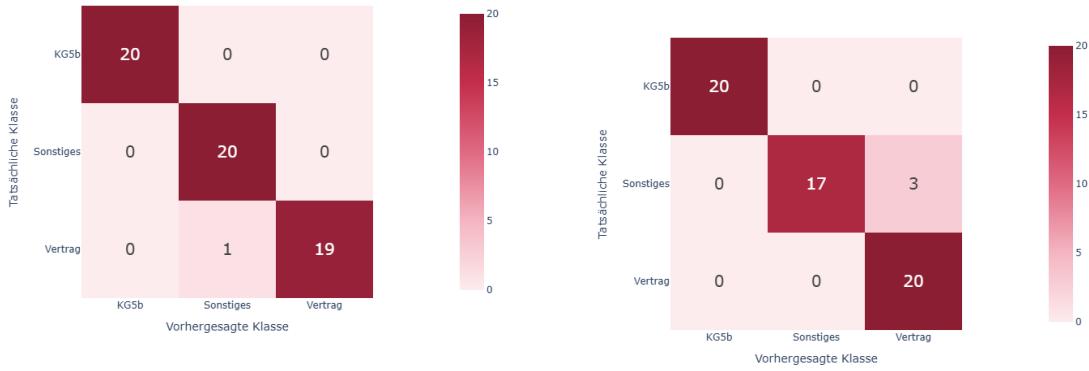
Die Latenz des großen Modells ist erkennbar größer als die des Qwen-2.5-VL-7B-finetuned. Wie in der Abbildung 4.8 zu erkennen, ist die Dauer der Inferenz des Qwen-2.5-VL-32B merklich höher. Mit einer maximalen Latenz von 252,06 Sekunden liegt das Qwen-2.5-VL-32B deutlich über dem Qwen-2.5-VL-7B-finetuned mit 21,6 Sekunden.

### 4.3 Ressourcenverbrauch der VLMs gegenüber der YOLO/OCR-Pipeline

Für den Vergleich des Ressourcenverbrauchs wird sowohl die VRAM-Auslastung als auch der Energieverbrauch betrachtet.

In Abbildung 4.9 ist die statische VRAM-Auslastung abgebildet. Die niedrigste Auslastung hat die YOLO/OCR-Pipeline mit 4,21 GB. Das Qwen-2.5-VL-7B und Qwen-2.5-VL-7B-finetuned befinden sich 16,4 GB und 16,88 GB auf einer Ebene. Während das Pixtral-12B mit 23,89 GB knapp über den 7B Modellen liegt, ist das Qwen-2.5-VL-32B mit 64,18 GB mit Abstand das größte Modell.

Die dynamische VRAM-Auslastung in Abbildung 4.10 zeigt ein anderes Ergebnis, jedoch liegt auch hier die YOLO/OCR-Pipeline mit einer durchschnittlichen Auslastung von 0,49 GB und einer maximalen von 1,02 GB unter den Mitstreitern. Interessant sind die Werte des Qwen-2.5-VL-7B-finetuned und des Qwen-2.5-VL-32B, denn hier liegt das kleinere Modell mit durchschnittlich 3,03 GB über dem großen Modell mit durchschnittlich 3,01 GB.



(a) Confusion Matrix für das Qwen-2.5-VL-7B-finetuned

(b) Confusion Matrix für das Qwen-2.5-VL-32B

Abbildung 4.7: Confusion Matrices aller Modelle

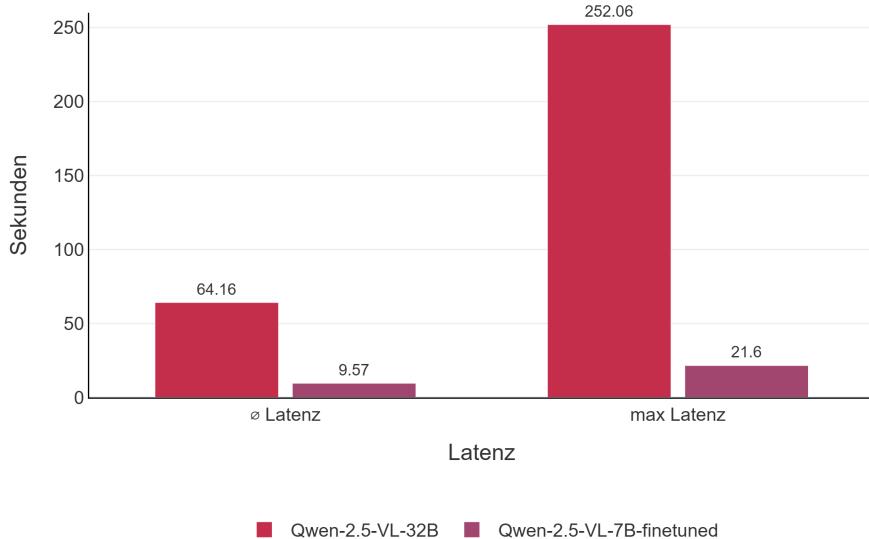


Abbildung 4.8: Ergebnisse der Latenzmessung

Der Energieverbrauch in Abbildung 4.11 zeigt den markantesten Unterschied zwischen der YOLO/OCR-Pipeline und den VLMs. Während die Pipeline nur 117 J im Durchschnitt verbraucht, benötigt selbst das effizienteste Modell, das Qwen-2.5-VL-7B, etwas das 22-fache an Energie. Im Vergleich zum Qwen-2.5-VL-32B, ist die Pipeline bei dem maximalen Verbrauch rund 250-fach sparsamer.

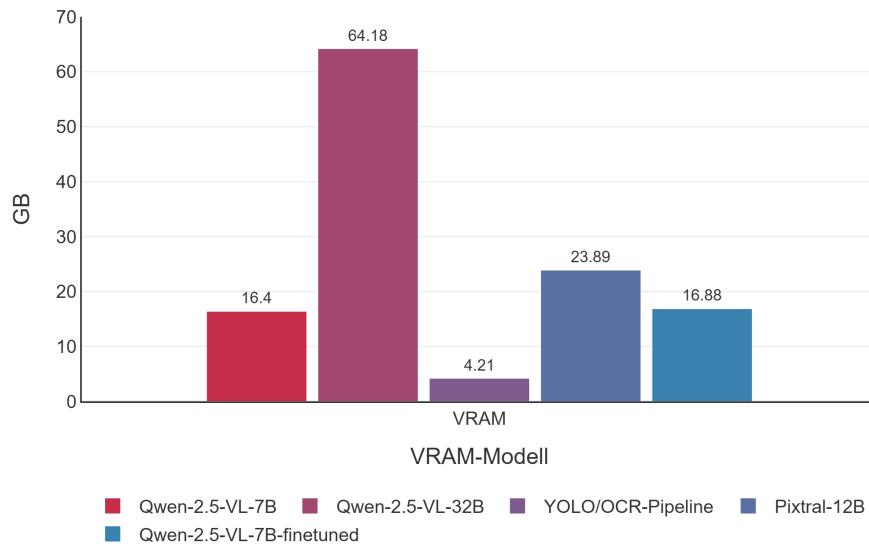


Abbildung 4.9: Ergebnisse der VRAM-Nutzung des geladenen Modells

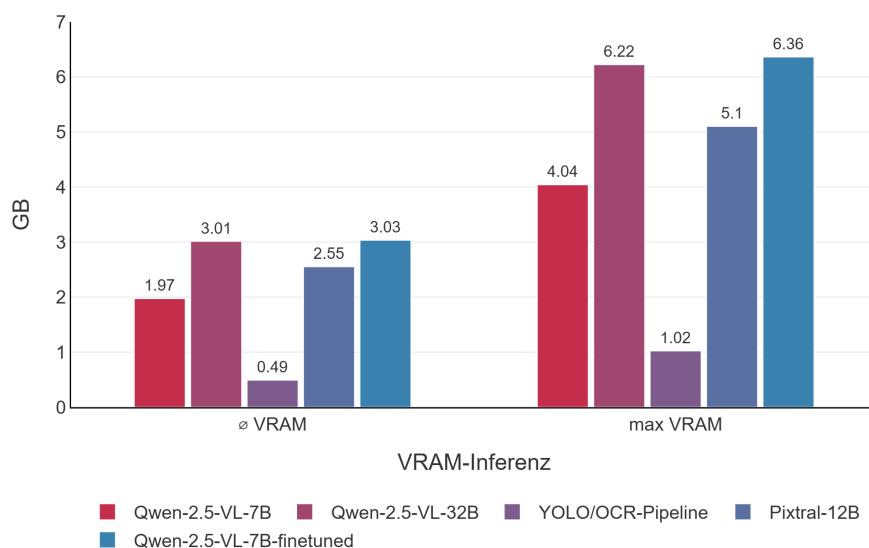


Abbildung 4.10: Ergebnisse der VRAM-Nutzung während der Inferenz

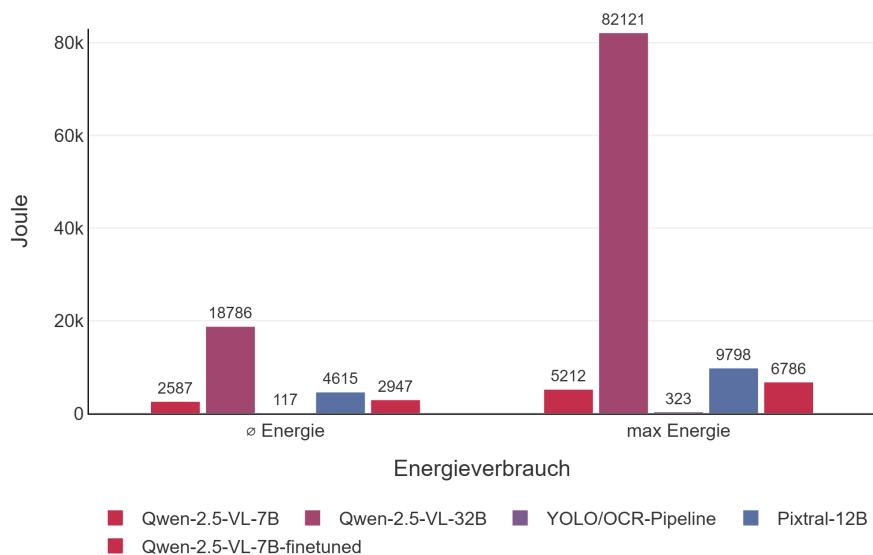


Abbildung 4.11: Ergebnisse des Energieverbrauchs

# Kapitel 5

## Diskussion

Eine besondere Herausforderung bei der Verarbeitung stellen die binären Informationen dar, wie etwa die Checkbox für das Feld `apprenticeship-ended`. Hier muss das Modell visuell unterscheiden, ob ein Kasten leer, angekreuzt oder durchgestrichen ist. Ebenso kritisch ist die Detektion von `signature_company` und `stamp_company`. Im Gegensatz zu Textfeldern ist hier nicht der textliche Inhalt des Stempels relevant, sondern lediglich dessen Vorhandensein. Die Stempel sind oft blass oder eingefärbt, was besonders bei Schwarz-Weiß Scans zu einer schlechten Qualität führt. Zudem liegen die Felder `date_document` und `signature_company` nah beieinander. Unterschriften sind regelmäßig größer als ihr vorgesehener Platz, wodurch das Feld `date_document` überdeckt wird. Zusätzlich werden die Stempel häufig zusammen mit der Unterschrift auf der linken Seite platziert.

- yolo ocr schneidet am besten ab wegen pipeline design bei sonstigen dokumenten - fehlerhafte klassifikation zieht sich durch - confusion matrix yolo ocr viele als sonstige -> große Fehler - kg5d - hohen Verbrauch erklären



# Kapitel 6

## Zusammenfassung

- constrained decoding



# Abbildungsverzeichnis

2.1	KG5b-Formular . . . . .	4
2.2	Ausbildungsvertrag der Industrie- und Handelskammer . . . . .	5
2.3	Schematischer Workflow der aktuellen AuBe-Pipeline . . . . .	5
2.4	Schematische Darstellung der Trainings- und Inferenzinfrastruktur . . . . .	6
2.5	Architektur eines Vision Language Models . . . . .	7
3.1	Überblick der Methodik . . . . .	11
4.1	Ergebnisse der Information Extraction . . . . .	26
4.2	Ergebnisse der Klassifikation . . . . .	26
4.3	Confusion Matrices aller Modelle . . . . .	27
4.4	Ergebnisse der Latenzmessung . . . . .	28
4.5	Ergebnisse der Information Extraction . . . . .	28
4.6	Ergebnisse der Klassifikation . . . . .	29
4.7	Confusion Matrices aller Modelle . . . . .	30
4.8	Ergebnisse der Latenzmessung . . . . .	30
4.9	Ergebnisse der VRAM-Nutzung des geladenen Modells . . . . .	31
4.10	Ergebnisse der VRAM-Nutzung während der Inferenz . . . . .	31
4.11	Ergebnisse des Energieverbrauchs . . . . .	32



## **Tabellenverzeichnis**

2.1 Benchmark-Ergebnisse der evaluierten Modelle[Bai 24, Qwen 25a, Agra 24] . . . . . 9



# **Listingverzeichnis**

1	Preprocessing der Dokumente . . . . .	12
2	JSON-Schema des Dokumententyps KG5b . . . . .	13
3	JSON-Schema des Dokumententyps Ausbildungsvertrag . . . . .	14
4	JSON-Schema des Dokumententyps Sonstiges . . . . .	14
5	Basisklasse der verschiedenen Comparator . . . . .	16
6	Comparator für Namen . . . . .	16
7	Konfiguration des KG5b-Formulars . . . . .	17
8	Ergebnis-Dictionary mit den Metadaten des gesamten Dokumentenkorpus . . . . .	18
9	Parameter, der bestimmt, ob Halluzinationen bestraft werden . . . . .	19
10	Finaler Prompt der Modellauswahl . . . . .	21
11	OpenAI Message Format . . . . .	22
12	Prompt für das Fine-Tuning des Qwen-2.5-VL-7B . . . . .	23
13	LoRA-Konfiguration . . . . .	24
14	Trainingsparameter des Fine-Tunings . . . . .	24



## Literaturverzeichnis

- [Agra 24] P. Agrawal *et al.* “Pixtral 12B”. 2024.
- [Arbe 22] B. für Arbeit. “Erklärung zum Ausbildungsverhältnis (KG 5b)”. [https://www.arbeitsagentur.de/datei/dok\\_ba031910.pdf](https://www.arbeitsagentur.de/datei/dok_ba031910.pdf), 2022.
- [Arbe 24] B. für Arbeit. “Kindergeld / Kinderzuschlag Jahreszahlen 2024”. [https://statistik.arbeitsagentur.de/Statistikdaten/Detail/202412/famka/famka-jz/famka-jz-d-0-202412-pdf.pdf?\\_\\_blob=publicationFile&v=3](https://statistik.arbeitsagentur.de/Statistikdaten/Detail/202412/famka/famka-jz/famka-jz-d-0-202412-pdf.pdf?__blob=publicationFile&v=3), 2024.
- [Bai 24] S. Bai *et al.* “Qwen2.5-VL Technical Report”. 2024.
- [Hand 25] D. I. und Handelskammer. “Berufsausbildungsvertrag”. <https://www.dihk.de/resource/blob/140492/9f4fa2617d668aa62f5d3caad77cdbaa/bildung-musterausbildungsvertrag-data.pdf>, 2025.
- [Hu 21] E. J. Hu *et al.* “LoRA: Low-Rank Adaptation of Large Language Models”. 2021.
- [Huan 21] Z. Huang *et al.* “ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction”. 2021.
- [Kala 23] D. Kalajdzievski. “A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA”. 2023.
- [Mang 22] S. Mangrulkar *et al.* “PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods”. <https://github.com/huggingface/peft>, 2022.
- [Mine 21] M. Minesh, K. Dimosthenis, and J. C.V. “DocVQA: A Dataset for VQA on Document Images”. 2021.
- [NVID 22] NVIDIA. “NVIDIA A40”. <https://www.nvidia.com/de-de/data-center/a40/>, 2022.
- [Pixt 24] Pixtral. “Pixtral-12B-2409”. <https://huggingface.co/mistralai/Pixtral-12B-2409>, 2024.
- [Qwen 25a] Qwen. “Qwen2.5VL-32B-instruct”. <https://huggingface.co/Qwen/Qwen2.5-VL-32B-Instruct>, 2025.

[Qwen 25b] Qwen. “Qwen2.5VL-7B-instruct”. <https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct>, 2025.

# Glossar

**Accuracy** Metrik, die den Anteil der insgesamt korrekten Vorhersagen an der Gesamtzahl der getroffenen Vorhersagen beschreibt.. i

**AuBe** Ausbildungsbescheinigungen. i

**BA** Bundesagentur für Arbeit. i, 1

**EXIF** Exchangeable Image File Format. i, 12

**F1-Score** Das harmonische Mittel aus Precision und Recall.. i

**FamKa** Familienkasse der Bundesagentur für Arbeit. i

**Few-Shot Prompting** Ein Ansatz, bei dem dem Modell mehrere Beispiele im Prompt übergeben werden.. i

**FN** False Negative. i, 18

**FP** False Positive. i, 18

**IDE** Integrated Development Environment. i, 7

**IE** Information Extraction (IE) umfasst nach Jurafsky und Martin die Aufgabe, Ereignisse oder Situationen in Dokumenten zu identifizieren und die entsprechenden Felder eines vorgegebenen Templates zu füllen[?]. Ziel hierbei ist es, vordefinierte Entitäten aus einem Dokumentenbild zu extrahieren und diese in ein maschinenlesbares Schema zu überführen.. i, 13, 14, 17, 19, 20, 25, 27, 45

**JSON** JavaScript Object Notation. i, 13, 14, 15, 17, 18, 20, 22

**Latenz** Die Zeitdauer, die das System für die Inferenz einer einzigen Antwort benötigt.. i, 11, 19, 26, 29

**Levenshtein-Similarity** Die Levenshtein-Similarity ist ein aus der Levenshtein-Distanz abgeleiteter, normalisierter Ähnlichkeitswert zwischen zwei Zeichenketten. Formal basiert sie auf der Levenshtein-Distanz, also der minimalen Anzahl von Einfügungen, Löschungen oder Ersetzungen, die nötig sind, um eine Zeichenkette in eine andere zu überführen[?]. i

**LLM** Large Language Model. i, 7, 8

**LoRA** Low-Rank Adaptation. i, 9, 22

**MLLM** Multimodal Large Language Model. i

**OCR** Optical Character Recognition. i, 1, 4, 5, 11, 16

**One-Shot Prompting** Eine Technik, bei der dem Modell genau ein Beispiel für die Lösung im Prompt bereitgestellt wird.. i

**OpenAI Message Format** Das **OpenAI Message Format** ist ein standardisiertes, rollenbasiertes Datenformat zur Interaktion mit Modellen, bei dem Nachrichten nach Rollen (z. B. `user`, `assistant`) unterteilt und Inhalte als Array aus verschiedenen Typen wie Text und Bild definiert werden.. i, 22

**PEFT** Parameter-Efficient Fine-Tuning. i, 9, 22

**Precision** Gibt den Anteil der tatsächlich positiven Fälle an allen vom Modell als positiv klassifizierten Fällen an.. i

**Prompt Engineering** Ist ein Prozess, um die effektivste Anweisung für eine spezifische Aufgabe zu finden[?]. i

**R-K-F-Formel** Strategie im Prompt Engineering zur strukturierten Anweisung eines LLMs. Dabei steht R für Rolle (Rolle, die das Modell einnehmen soll), K für Kontext (relevante Hintergrundinformationen) und F für Format (Vorgabe der gewünschten Ausgabestruktur).. i

**Recall** Gibt den Anteil der korrekt als positiv erkannten Fälle an allen tatsächlich vorhandenen positiven Fällen an.. i

**rsLoRA** Rank-Stabilized Low-Rank Adaptation. i, 9, 22

**TN** True Negative. i, 18

**TP** True Positive. i, 10, 18

**Unsloth** Open-Source-Framework zum beschleunigten und speichereffizienten Fine-Tuning von Large Models, das speziell für Methoden des PEFT optimiert ist.. i, 22

**ViT** Vision Transformer. i, 8

**VLM** Vision Language Model. i, 1, 2, 7, 8, 11, 13, 15, 17, 22, 23, 25, 26, 30

**VQA** Visual Question Answering. i, 9

**VRAM** Video Random Access Memory. i, 7, 8, 11, 19, 22, 29

**VRAM-Auslastung** Der maximale Bedarf an Grafikspeicher der GPU, der während der Ausführung eines Modells gemessen wird.. i

**YOLO** You Only Look Once. i, 1, 4, 6, 11, 16

**Zero-Shot Prompting** Eine Methode, bei der dem Modell eine Aufgabe gestellt wird, ohne dass zusätzliche Beispiele im Prompt enthalten sind.. i