Fachhochschule Südwestfalen

Echtzeit-Interaktion auf Smartglasses durch latenzarme Sprachverarbeitung und datenschutzwahrende Gesichtserkennung



Kerem Deniz Yasar

Referent: Dr.-Ing. Stefan Goetze

Korreferent: Benjamin Diez

Masterarbeit zur Erlangung des akademischen Grades Master of Science im Verbundstudiengang Angewandte Künstliche Intelligenz

in der Fachabteilung

Informatik und Naturwissenschaften

9. Januar 2026

Abstract

Mit dem Einbau von technischen Komponenten in sogenannte Smart Wearables wird die Integration von Alltagsassistenten immer einfacher. Ziel dieser Arbeit ist die Implementierung eines solchen Assistenten, der Zugriff auf Mikrofon und Kamera hat. Der Fokus besteht hierbei insbesondere auf der Wahrung einer latenzarmen Interaktion bei Sprachinteraktionen und der Wahrung der Privatsphäre bei der Bildverarbeitung. Die damit zu untersuchenden Fragen beschäftigen sich demnach mit den technischen Voraussetzungen für die Datenpipeline, den nötigen Mitteln zur Latenzminimierung und einer performanceoptimierten Bildverarbeitungsmethodik.
Zur näheren Betrachtung wurden neben den technischen Voraussetzungen verschiedene Methoden, Modelle und Schnittstellen getestet, die sich alle als Teilkomponenten im Gesamtsystem wiederfinden lassen. Neben der automatischen Spracherkennung und -Synthese ist die Sprachgenerierung ein fundamentaler Bestandteil. Die damit einhergehende Bildverarbeitung unterteilt sich in Face-Detection und -Recognition. Für all diese Modelle ist eine Betrachtung relevanter Metriken nötig, um Aufschluss über den Erfolg und die Vergleichbarkeit zwischen den jeweiligen Methoden ziehen zu können.
Aufgrund der derzeitigen Größe von Large Language Models ist deren Nutzung über offizielle Schnittstellen weitaus vorteilhafter als ein lokaler Betrieb. Dies spiegelt sich vor allem in der Latenzzeit wider. Hierzu hat sich Geminis 2.5 Flash-Lite mit einer Antwortzeit von 1,1 Sekunden als ideales Element erwiesen. Für die automatische Spracherkennung hat das Faster Whisper Medium-Modell aufgrund seines Trade-offs zwischen Genauigkeit, Geschwindigkeit und Skalierbarkeit in vollem Maße überzeugt. Dabei lag die durchschnittliche Transkriptionszeit bei nur 0,152 Sekunden. Die Sprachsynthese basiert im praktischen Teil auf FastPitch, welches neben ungeschlagener Geschwindigkeit auch authentisch generierte Stimmen ermöglicht. Bei der Bildverarbeitung gab es aus Sicht der Metriken Accuracy, Precision und Recall ein enges Rennen zwischen den Yolov8n-Face- und ArcFace pretrained Buffalo-Modellen. Doch mit Blick auf die Gesichtserkennung, in der sich die Buffalo SC-Variante als optimal für latenzarme Systeme erwiesen hat, war eine Entscheidung zugunsten Letzterer für beide Zwecke im Hinblick auf die Gesamtarchitektur schlüssig.
Die vorgestellten Modelle und Methodiken bestätigten den SOTA-Ansatz für ihre jeweiligen Zwecke. Die Implementierung im praktischen Teil wies geringe Latenzzeiten in der Sprachverarbeitung von unter einer Sekunde und einem RTF von 0,1 auf. Auch die Bildverarbeitung verlief optimal und ließ sich mit über 100 Frames per Second nicht nur ohne Probleme gezielt ansteuern, sondern auch untersuchen.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation und Problemstellung . . . 1
- 1.2 Problemstellung und Herausforderung . . . 2
- 1.3 Forschungsfragen und Methodik . . . 2
- 1.4 Aufbau der Arbeit . . . 3

2 Grundwissen und verwandte Arbeit **4**
- 2.1 Ray-Ban Meta Glasses . . . 4
- 2.2 Automatisierung von Messaging-Systemen . . . 5
- 2.3 Large Language Modelle und Googles Generative AI . . . 7
  - 2.3.1 Transformer-Architektur . . . 7
  - 2.3.2 State-of-the-Art LLM-Familien und Googles Gemini AI . . . 9
  - 2.3.3 API . . . 10
- 2.4 Grundlagen der Sprachverarbeitung: Speech-to-Text und Text-to-Speech . . . 11
  - 2.4.1 Speech-to-Text (STT) . . . 11
  - 2.4.2 Text-to-Speech (TTS) . . . 14
- 2.5 Grundlagen der automatisierten Gesichtsverarbeitung . . . 18
  - 2.5.1 Verfahren zur Gesichtsdetektion . . . 19
  - 2.5.2 Verfahren zur Gesichtserkennung . . . 23
  - 2.5.3 Die Bedeutung vortrainierter Modelle . . . 25
- 2.6 Egocentric Vision Systeme und KI-Assistenten . . . 26
  - 2.6.1 Einführung in die egozentrische Vision und Datenerfassung . . . 26
  - 2.6.2 Die Bedeutung qualitativ hochwertiger Datensätze . . . 27
  - 2.6.3 Forschung und Implementierungen von egozentrischen KI-Assistentendiensten 27
  - 2.6.4 Zusammenfassungstechniken für egozentrische Video- und Audiodaten 29
  - 2.6.5 Gesichtserkennungsalgorithmen und Anwendungen . . . 29
  - 2.6.6 Integration von Sprachbefehlen zur Steuerung von Assistenzdiensten . 30
  - 2.6.7 Identifizierte Forschungslücken und zukünftige Richtungen . . . 31

3 Implementierung und Methodik **32**
- 3.1 Systemarchitektur . . . 32
- 3.2 Entwicklungsumgebung und eingesetzte Technologien . . . 33
- 3.3 Browser Automatisierung . . . 35
- 3.4 Chatbot . . . 39
- 3.5 Sprachverarbeitung . . . 40

3.6 Gesichtsverarbeitung 44
3.7 Ablaufsteuerung und Exekution 49

4 Evaluation und Ergebnisse **50**
4.1 Evaluationsmethodik 50
4.1.1 Datensätze 50
4.1.2 Evaluationsmetriken 53
4.2 Ergebnisse der Teilkomponenten 57
4.2.1 LLM 57
4.2.2 Speech-to-Text 61
4.2.3 Text-to-Speech 63
4.2.4 Face Detection 66
4.2.5 Face Recognition 68
4.3 Evaluation und Diskussion der Ergebnisse 72

5 Diskussion **75**
5.1 Interpretation und Einordnung der Arbeit 75
5.1.1 Ergebnisse 75
5.1.2 Rückblick auf die Forschungsfragen 75
5.2 Kritische Reflexion und Herausforderungen 76
5.3 Einordnung in den Stand der Technik 78
5.4 Limitierungen der Arbeit 79

6 Fazit und Ausblick **81**
6.1 Zusammenfassung der Arbeit 81
6.2 Beantwortung der Forschungsfragen 82
6.3 Ausblick auf zukünftige Arbeiten 82

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

Mit einem Blick auf die Vergangenheit ergeben sich deutliche Unterschiede in der Komfortabilität der genutzten technischen Produkte und Technologien. Selbstverständliche Komponenten des aktuellen Alltags waren zuvor nur in Filmen, Serien oder Sci-Fi-Büchern gegeben. Während die kabellose Echtzeit-Kommunikation Mitte des 20. Jahrhunderts weit entfernt war, kam 1987 bereits der erste Schritt in Richtung Konsumentennutzung und ist heutzutage nicht mehr wegzudenken [1]. Auch Kopfhörer hatten kurz danach einen Durchbruch und ermöglichten es bereits Anfang der 2000er, ohne Kabel Unterhaltungen und Ton zu übertragen. Ohne Bluetooth war das zu der Zeit davor unvorstellbar [2]. Der Startschuss war gegeben, und Videotelefonate, Tablets und autonome Fahrzeuge waren nun visuell als Ziel einer technologischen Zukunft greifbar nahe. Nicht zuletzt mit der Markteinführung des iPhones waren die vorerst letzten kritischen Einwände gegenüber der technischen Möglichkeit von Implementierung und Anwenderfreundlichkeit gebrochen. Interaktive Displays, gepaart mit intelligenten Funktionen, sollten den Konsumenten die Bedienung so leicht wie möglich machen. Zugleich kam damit die Frage auf, auf welche Art und Weise Interaktion und Erlebnis vertrauensvoller werden würden. Fortschritte in der Künstlichen Intelligenz und den damit verbundenen Sprachassistenten brachten einen signifikanten Sprung in Richtung Mensch-Maschinen-Kommunikation. Die ersten Sprach-zu-Sprache-Pipelines waren noch unvertraut, zeigten jedoch früh das Potenzial im Bereich Unterhaltung und Informationsgewinn. Neben auditiver Unterstützung und Unterhaltung gab es nun also noch das Feld der virtuellen Realität. Erste Ideen zur Anwendung kamen Ende der 80er und 90er unter Forschern aus verschiedenen Branchen auf, darunter NASA und Boeing [3] [4]. Es wurde früh das Potenzial einer virtuellen Realität erkannt, die in vielen Bereichen Anwendungsmöglichkeiten gezeigt hat. Nicht zuletzt kam das Branding von Facebook sogar zu Meta, welches als Begriff bereits einen Ausblick auf die Metawelt geben sollte. Die Meta Quest dient auch heute noch als Vorzeigeprodukt im Bereich Augmented Reality und hat im Markt deutlichen Zuwachs an Wettbewerbern erhalten, darunter auch Apple's Vision Pro [5].

Der nächste Schritt im Rennen um die Einbindung von Technik in den Alltag ist klar: Tragbare Smart-Devices, deren Zweck hauptsächlich in der künstlichen Unterstützung der Träger liegt und die durch sprach- oder gestenbasierte Steuerung angetrieben werden. Neben Smartwatches sind Metas Ray-Ban Smart Glass hierbei ein ideales Beispiel. Mit der Fähigkeit,

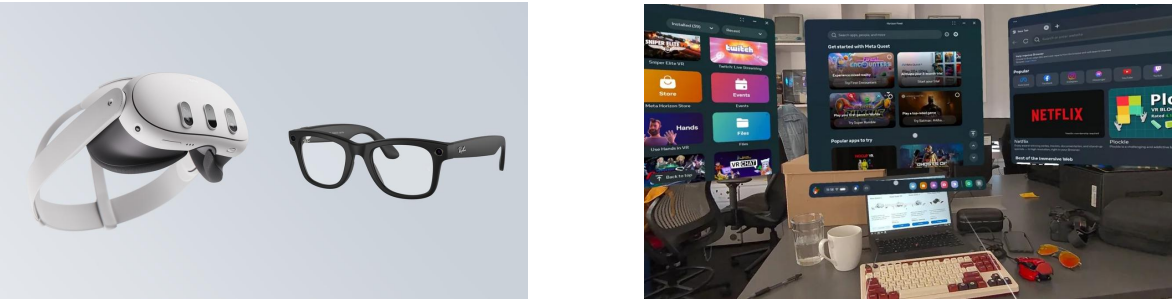


Figure 1.1: *Meta Quest 3 und die Ray-ban Meta Glasses [5] mit Quests Displaysicht [6]*

auf Abruf zu kommunizieren und aufzunehmen, ist der Konsument jederzeit digitalisiert. Unzählige Firmen haben sich dem Trend angeschlossen und gehören nun zu den Pionieren im Bereich Smart Wearables. Dazu zählen nicht nur Meta und Google, sondern auch eine Reihe von Unternehmen wie XREAL, RayNeo, INMO, Amazon und viele weitere [6].

1.2 Problemstellung und Herausforderung

Genau an diesem Punkt kommen jedoch mehrere Fragen auf. Welche Bedingungen müssen erfüllt sein, um die Einbindung in den Alltag zu standardisieren, sodass später ein Szenario ohne Brille undenkbar wäre? Dazu muss neben der Handhabung und dem Wissensschatz sichergestellt werden, dass die Qualität und Relevanz des Feedbacks maßgeblich für den kommerziellen Erfolg sind. Dies kann in verschiedenen Formen geschehen, muss dem Verbraucher jedoch jederzeit die Verfügbarkeit garantieren. Darüber hinaus ist die Integration eines Mikrofons und einer qualitativ hochwertigen Kamera im ersten Moment interessant, weckt jedoch das negative Interesse unterschiedlichster Parteien. Durch die Verarbeitung personenbezogener Daten geraten diese Geräte in kritische Gebiete, die gerade in der Europäischen Union immer wieder zu Problemen führen. Eine der größten Hürden stellt hierbei die Datenschutzgrundverordnung dar [5]. Dementsprechend müssen, unter Berücksichtigung menschlichen subjektiven Feedbacks und der Gesetze, Lösungen konstruiert und implementiert werden, die diesem entgegenwirken und als nahezu ideale Lösung wirken.
Wie in späteren Kapiteln zu sehen sein wird, fokussiert sich die aktuelle Forschung anhand von Wearable Devices und dem sogenannten Ego Corpus vor allem auf Aufgaben, die der Assistenz in alltäglichen Situationen dienen. Performance und Datenschutz stehen hierbei aufgrund der im Hintergrund laufenden Dienste und Ursprungsorte wie den USA nicht im Vordergrund, weshalb diese im Folgenden besonders im Fokus stehen.

1.3 Forschungsfragen und Methodik

Das Ziel wird es sein, ein vollumfängliches Projekt zu bauen, das mittels gegebener Hardware und zugänglicher Software in Echtzeit Daten verarbeiten und wiedergeben kann. Dazu zählt nicht nur die auditive Verarbeitung von Sprache und Text, sondern auch die visuelle Analyse des gegebenen Kameramaterials. Um die zuvor erhobenen Punkte zu untersuchen, müssen vorab Fragen aufgestellt werden, die beantwortet und diskutiert werden sollen.

Wie lässt sich eine zuverlässige Pipeline aufstellen, mit der kontinuierlich Daten sowohl gesammelt als auch gesendet werden können? Dieser Schritt behandelt die Grundlage der Forschungsarbeit, ohne die kein nennenswerter Fluss zustande kommen könnte. Für die Arbeit wird die Meta Ray-Ban-Brille mittels Automatisierungssoftware in Python indirekt gekoppelt und dient als maßgebliche Sicht und Quelle.
Inwieweit steuern Speech-to-Text und Text-to-Speech Modelle das Erlebnis in Bezug auf die Interaktion? Für eine reibungslose Kommunikation sind Sprachmodelle in jeder Hinsicht unerlässlich. Dabei ist es jedoch nicht nur wichtig, auf die Architektur und den Inhalt einzugehen, sondern auch auf die dynamische Aktivität unter Berücksichtigung der Latenz zwischen Input und Output. Zur Untersuchung optimaler und effizienter Methoden werden aktuelle und populäre Modelle genauer betrachtet und gegeneinander unter Berücksichtigung verschiedener Metriken verglichen. Damit soll die Wahl der Komponenten in der finalen Software genauer verdeutlicht werden.
Wie kann Gesichtserkennung unter Wahrung der Privatsphäre realisiert werden? Neben dem Datenfluss zwischen Verbraucher und Software soll untersucht werden, wie der Umgang mit personenbezogenen Daten gestaltet werden muss, um die zuvor genannten Probleme zu umgehen und nachhaltig konfliktfrei zu lösen.

1.4 Aufbau der Arbeit

In Kapitel 2 erfolgt zunächst eine fundierte Darlegung der zugrunde liegenden Konzepte und Systemgrundlagen. Dies umfasst nicht nur die Selektion von Hard- und Softwarekomponenten, sondern auch ein tiefgreifendes Verständnis der angewandten Modelle und Architekturen. Letztere stellen den essentiellen Kern des vorliegenden Projekts dar und sind für die Validierung der Ergebnisse in den Benchmark-Tests von immanenter Bedeutung. Das nachfolgende Kapitel widmet sich detailliert der Systemarchitektur. Dabei wird nicht nur präzise ausgeführt, aus welchen individuellen Komponenten der Datenfluss resultiert, sondern auch die ganzheitliche Funktionsweise des Systems umfassend dargestellt. In Kapitel 4 wird die Auswahl der Systembestandteile anhand diverser Benchmarks ausführlich erläutert und visuell aufbereitet. Dies umfasst sowohl die Analyse aktueller Methoden als auch eine kritische Evaluierung ihrer praktischen Anwendbarkeit. Abschließend werden in Kapitel 5 und 6 die gewonnenen Ergebnisse evaluiert und diskutiert. Dies schließt nicht nur die Präsentation von Lösungsansätzen, identifizierten Problemen und Herausforderungen ein, sondern auch einen Ausblick auf zukünftige Methodologien, deren Implementierung maßgeblich für einen signifikanten Fortschritt erachtet wird.

Kapitel 2

Grundwissen und verwandte Arbeit

2.1 Ray-Ban Meta Glasses

Die Auswahl an Smart Wearable Devices wird mit der Zeit nur größer. Wie zu Beginn erwähnt, ist das Wachstum gerade auch auf die Nachfrage und das Verlangen der Konsumenten zurückzuführen. Die Geräte unterscheiden sich hierbei leicht und zugleich stark voneinander. Abhängig vom Gerät wird besonderer Fokus auf spezifische Attribute gelegt. Dies können Übersetzungs- und Aufnahmefunktionen sein, jedoch parallel auch stilistische Aspekte, die die Integration in den Alltag so bequem wie möglich gestalten. Die Wahl der Ray-Ban Glasses von Meta ist in diesem Fall gefallen, da sie sich durch ihre technischen Funktionalitäten und die leichte Bedienung ideal für den vorgesehenen Zweck eignen.
Mit dem Ziel der unauffälligen Integration befindet sich an der Vorderseite eine 12-Megapixel-Ultraweitwinkel-Kamera, die mit Fotos im Portraitformat neben den Aufzeichnungen im 1080p einwandfreie Aufnahmen ermöglicht [7, 8]. Aufgrund der 154 mAh und des eingebauten Qualcomm Snapdragon AR1 Gen 1 Chips können Aufnahmen in der Regel nur bis zu 3 Minuten am Stück erstellt werden. Im Kontrast besteht jedoch die Möglichkeit eines (Video-)Anrufs oder auch Livestreams, die keine zeitlichen Vorgaben enthalten. Trotzdem muss achtsam mit der Überhitzung des Akkus umgegangen werden, der jederzeit im Ladecase aufgeladen werden kann und somit eine Laufzeit von bis zu 36 Stunden verspricht [8, 9].
Von entscheidender Bedeutung ist die Inklusion von Kamera und Lautsprecher. Die Sprachsteuerung, Telefonie oder Wiedergabe von Audioinhalten erfolgen mittels eines 5-Array-Mikrofons und offener Lautsprecher im Bügel. Der Ton wird dabei in Richtung Ohr gelenkt, ohne den Gehörgang zu blockieren [7, 8].
Die Balance zwischen Rechenleistung, Energieverbrauch und Akkulaufzeit ist eine der kritischsten Herausforderungen im Smart-Glasses-Design. Das Vorhandensein und die Art eines Displays definieren maßgeblich, ob es sich um eine reine "Connected Brilleöder um eine echte AR-Brille handelt. Die Ray-Ban Meta verzichtet bewusst auf eine AR-Displayfunktion. Ihr primärer Interaktionsmechanismus neben der Sprachsteuerung ist ein Touchpad am Brillenbügel. An Bord befinden sich zudem eine Inertialmesseinheit (IMU) zur Erfassung von Bewegungen und ein Näherungssensor, um zu erkennen, ob die Brille getragen wird [7].
Die Architektur des Datenflusses, von der Erfassung über die lokale Verarbeitung bis hin zur Übertragung und Cloud-Analyse, ist ein weiterer entscheidender Unterschied zwischen den Plattformen und spiegelt deren Einsatzzweck wider. Bei der Ray-Ban Meta ist der Datenfluss

auf Einfachheit und die Anbindung an das Smartphone-Ökosystem ausgelegt. Fotos, Videos und Audioaufnahmen werden zunächst direkt im 32 GB großen internen Speicher des Geräts abgelegt. Eine grundlegende Bild- und Audioverarbeitung findet ön-deviceßtatt, ebenso wie die KI-basierte Erkennung von Aktivierungswörtern für die Sprachsteuerung ("Hey Meta"). Für die Weiterverwendung werden die Medien mittels Bluetooth oder Wi-Fi 6 an die Meta View App auf dem gekoppelten Smartphone übertragen. Von dort können sie geteilt oder weiterbearbeitet werden. Livestreams werden direkt von der Brille über das Smartphone an die entsprechenden Social-Media-Plattformen gesendet [7, 8]. Anspruchsvollere KI-Funktionen und die vollständige Verarbeitung von Sprachbefehlen können je nach Komplexität entweder lokal auf dem Gerät oder über eine Anbindung an die Cloud-Infrastruktur von Meta erfolgen [10]. Die Ray-Ban Meta operiert als geschlossenes System. Es steht kein öffentliches SDK oder eine API für Drittentwickler zur Verfügung. Die Interaktionsmöglichkeiten sind auf die von Meta bereitgestellte View App und die vordefinierten Sprachbefehle beschränkt. Diese geschlossene Systemarchitektur verhindert, dass Entwickler eigene Anwendungen erstellen oder direkten Zugriff auf die Kamera- und Sensorstreams erhalten, um innovative Anwendungsfälle zu realisieren [11, 12]. Ein nicht-technisches, aber eng damit verknüpftes Thema ist der Datenschutz. Meta versucht, Bedenken durch sichtbare LED-Indikatoren bei Kameraaktivität zu begegnen. Dennoch werfen die kontinuierliche Aufzeichnung und Übertragung von Audio- und Videodaten, insbesondere im Kontext neuer KI-gestützter Analysen, grundlegende Fragen zur Privatsphäre auf, die durch entsprechende Datenschutzrichtlinien adressiert werden müssen [10].



Figure 2.1: *Technische Komponenten der Meta Glasses [13]*

2.2 Automatisierung von Messaging-Systemen

Als grundlegende Technologie zur Automatisierung des Datenflusses wird eine Programmiersprache sowie ein entsprechend passendes Framework benötigt.
Die primäre Wahl der zugrunde liegenden Programmiersprache fällt auf Python. Die Entscheidung beruht nicht nur auf pragmatischen Aspekten, sondern auch auf strategischen, aufgrund der gegebenen Vielzahl an Bibliotheken. Dazu zählen auch die in späteren Kapiteln

erwähnten Machine Learning Algorithmen, deren Implementierung vor allem durch Python ermöglicht werden kann [14, 15]. Gepaart mit der globalen Fülle an Dokumentation [16] kann in nur wenigen Zeilen Code Komplexität und Inhalt wiedergespiegelt werden [17]. Aufgrund der kontinuierlichen Entwicklung der Programmiersprache durch globale Akteure wie Google und Facebook kann gerade in Themen wie Deep Learning verlässlich mit Python gearbeitet werden [17].
Für die Interaktion mit modernen, dynamischen Webanwendungen ist ein Werkzeug erforderlich, das in der Lage ist, einen Webbrowser programmatisch zu steuern und Benutzerinteraktionen zu simulieren. Selenium etabliert sich hier als ein führendes Open-Source-Framework [18]. Es wird primär für die Testautomatisierung von Webanwendungen eingesetzt, ist aber ebenso hervorragend für die Automatisierung wiederkehrender Aufgaben und das Web-Scraping geeignet [18, 19]. Das Herzstück von Selenium ist der WebDriver [20, 21]. Dieser fungiert als eine Programmierschnittstelle, die eine sprachunabhängige Kommunikation zwischen dem Steuerungsskript, in diesem Fall Python, und dem nativen Automatisierungs-Support eines Webbrowsers ermöglicht [21, 22]. Für jeden unterstützten Browser (z.B. Chrome, Firefox, Edge) existiert ein spezifischer "Browser Driver"[22]. Dieser Treiber empfängt Befehle vom Selenium-Skript über das JSON Wire Protocol oder das neuere W3C WebDriver Protocol via HTTP und übersetzt sie in Aktionen, die der Browser ausführt [22, 23]. Diese Architektur erlaubt es, dieselben Skripte mit geringfügigen Anpassungen über verschiedene Browser und Betriebssysteme hinweg auszuführen [20]. Die Notwendigkeit für ein Werkzeug wie Selenium ergibt sich aus der Architektur moderner Webseiten wie Facebook. Diese sind keine statischen HTML-Dokumente mehr, sondern hochgradig dynamische Single-Page-Applications, die Inhalte mittels JavaScript nachladen, ohne dass die Seite vollständig neu geladen wird (AJAX) [19, 20]. Einfache HTTP-Anfragen, wie sie von Bibliotheken wie "Requestsäusgeführt werden, würden nur das initiale HTML-Grundgerüst erhalten. Selenium hingegen steuert einen vollwertigen Browser, der JavaScript ausführt und somit in der Lage ist, mit der vollständig gerenderten Seite zu interagieren, auf Schaltflächen zu klicken, Formulare auszufüllen und durch Inhalte zu scrollen [24, 25]. Dies ist für das vorliegende Projekt unabdingbar, um die Benutzeroberfläche von Facebook Messenger zu navigieren und die relevanten Daten zu extrahieren.
Zuletzt stellt sich die Frage, inwiefern diese Tools nötig sind. Meta besitzt über mehrere Schnittstellen, deren Gebrauch die Bedienung auf operativer Seite mit vielen Vorteilen verbunden ist [26, 27]. Aufgrund der Regulierung gegenüber Nicht-Geschäftskunden ist diese jedoch nicht nutzbar, weshalb eine eigene Lösung konstruiert werden muss [28, 29]. Diese Restriktion macht den Einsatz von Browser-Automatisierung mittels Selenium zur einzig gangbaren Methode für dieses Projekt [30, 31]. Anstatt eine offizielle Schnittstelle zu nutzen, emuliert der Ansatz das Verhalten eines menschlichen Nutzers, der sich über den Webbrowser in sein Konto einloggt und mit der Messenger-Oberfläche interagiert [30, 31]. Dieser Prozess umfasst die Automatisierung der folgenden Schritte:

1. **Login:** Das Selenium-Skript navigiert zur Facebook-Login-Seite und gibt die Anmeldeinformationen in die entsprechenden Formularfelder ein, um eine authentifizierte Sitzung zu starten [31, 32].

2. **Navigation:** Nach erfolgreicher Anmeldung navigiert das Skript zur Messenger-Ansicht.

3. **Interaktion und Datenextraktion:** Das Skript identifiziert die HTML-Elemente, die

die Konversationen und Nachrichten enthalten. Aufgrund der dynamischen Natur von Facebook, bei der Inhalte nachgeladen werden, während der Nutzer scrollt, muss das Skript Aktionen wie das Scrollen simulieren, um alle relevanten Daten im Document Object Model (DOM) sichtbar zu machen [24]. Anschließend werden die gewünschten Daten aus dem DOM extrahiert.

Dieser Ansatz birgt jedoch Herausforderungen. Facebook implementiert Mechanismen, um automatisierten Zugriff zu erschweren, wie zum Beispiel sich dynamisch ändernde Klassennamen und IDs von HTML-Elementen oder die Erkennung von bot-ähnlichem Verhalten, was zu temporären Sperren oder zusätzlichen Sicherheitsabfragen, wie der 2-Faktor-Authentifizierung, führen kann [24, 33]. Daher erfordert die Implementierung robuste Selektoren zur Elementidentifikation und Strategien zur Nachahmung menschlichen Verhaltens, zum Beispiel durch Pausen zwischen Aktionen. [30].

2.3 Large Language Modelle und Googles Generative AI

Eine revolutionäre Wende hat die Entwicklung von Sprachmodellen und großen Sprachmodellen (LLMs) in den letzten Jahren im Bereich der künstlichen Intelligenz erweckt. Nicht zuletzt mit ChatGPT haben sich Vorteile und Einfluss auch im alltäglichen Gebrauch wirksam gezeigt. Als Grundlage für komplexe Dialogsysteme soll näher erläutert werden, welcher technische Aufbau sich dahinter verbirgt und inwiefern dieser dem vorliegenden Projekt substantiellen Mehrwert liefert.
Sprachmodelle sind als Systeme vor allem für das Verständnis und die Generierung menschlicher Sprache zuständig [34]. Statistische Muster und semantische Beziehungen zwischen Wörtern und Konzepten werden dabei auf Grundlage enormer Textmengen erlernt und eingeprägt. Mithilfe dieser werden Wortwahrscheinlichkeiten betrachtet und in einer finalen Ausgabe so wiedergegeben, dass der Output dem eines Menschen ähnelt. Erste Ansätze gab es bereits in den 1950er Jahren [34], waren jedoch eher als regelbasierte Systeme ausgelegt, die auf linguistischen Regeln basierten. Erst die Einführung statistischer Ansätze, wie unter anderem den N-Gramm-Modellen, hat dazu geführt, dass man sich der Komplexität der menschlichen Sprache genähert hat. Dabei wurden durch Worthäufigkeiten und Platzierungen erste Annäherungen versucht, um Wortsequenzen korrekt zu platzieren.
Die neueste Generation an Sprachmodellen sind die sogenannten Large Language Modelle. Durch beeindruckende Leistungsfähigkeit, basierend auf außergewöhnlicher Größe, schaffen es diese Modelle, Zusammenhänge in großem Maßstab einzufangen und ohne Probleme wiederzugeben [35]. Diese Modelle sind transformer-basierte neuronale Netzwerke mit Milliarden von Parametern, die auf riesigen Textkorpora trainiert wurden [36]. Durch Muster in den Trainingsdaten können damit ohne Probleme regelbasierte Zusammenhänge aufgestellt werden [37]. Damit zeigt sich, inwiefern diese Modelle "largeßind. Nicht nur die Anzahl der Parameter ist dabei gemeint, sondern auch die Milliarden von verwendeten Tokens [35]. Mit Hilfe von selbstüberwachtem Lernen wird gelernt, Text vorherzusagen.

2.3.1 Transformer-Architektur

Die Sprachmodellierung wurde im Paper Ättention Is All You Need"mit Hilfe der Transformer-Architektur von Grund auf revolutioniert [38]. Mit dem Verzicht auf die üblicherweise ge-

nutzten rekurrenten oder konvolutionalen Schichten basiert diese Architektur nun primär auf Aufmerksamkeitsmechanismen [39]. Der Kern besteht hierbei aus mehreren Komponenten.

- **Encoder-Decoder-Struktur**: Die ursprüngliche Transformer-Architektur folgt einem Encoder-Decoder-Paradigma, wobei der Encoder die Eingabe verarbeitet und der Decoder die Ausgabe generiert. Moderne LLMs verwenden häufig nur den Decoder-Teil (decoder-only models), wie bei der GPT-Familie [39].
- **Self-Attention-Mechanismus**: Der Self-Attention-Mechanismus bildet das fundamentale Element der Transformer-Architektur. Er ermöglicht es dem Modell, die wechselseitigen Beziehungen zwischen sämtlichen Wörtern innerhalb eines Satzes simultan zu erfassen. Dieser Mechanismus bestimmt für jedes Wort eine gewichtete Summe aller anderen Kontextelemente, wobei die vergebenen Gewichte die Bedeutsamkeit jedes Wortes für das jeweils betrachtete Wort abbilden [40][41].
- **Multi-Head Attention**: Für die Erfassung der Beziehungen verschiedener Wörter werden in Transformern *Multi-Head Attention* eingesetzt. Dabei werden mehrere Attention-Köpfe parallel betrieben, von denen jeder verschiedene Aspekte der Eingabe fokussiert [42].
- **Positionskodierung**: Um die Position von Wörtern in einer Sequenz für Transformer-Modelle kenntlich zu machen, werden Positionskodierungen (Positional Encodings) eingesetzt. Dies ist erforderlich, weil Transformatoren einen eingebauten Mechanismus zur Erfassung von Reihenfolgeinformationen fehlt [42].

Im nachfolgenden Bild lassen sich Reihenfolge und Anwendung der erwähnten Komponenten erkennen:

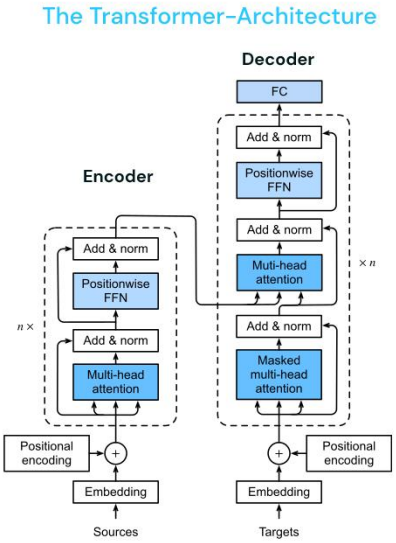


Figure 2.2: *Prinzipieller Aufbau einer Transformer-Architektur in Large Language Modellen [43]*

Unter den aktuellen LLM-Architekturen basieren die meisten überwiegend auf der Transformer-Technologie, haben jedoch verschiedene Spezialisierungen entwickelt [36]. Decoder-only-Modelle wie GPT haben sich als besonders effektiv für generative Aufgaben erwiesen, während *Encoder-only-Modelle* wie BERT für Verstehensaufgaben optimiert sind [37, 44]. Das Training von LLMs erfolgt in mehreren Phasen. Zunächst wird ein *Pre-Training* auf riesigen, ungelabelten Textkorpora durchgeführt, bei dem das Modell lernt, das nächste Wort in einer Sequenz vorherzusagen [35]. Anschließend folgt oft ein *Fine-Tuning* auf spezifische Aufgaben oder Domänen. Moderne Ansätze verwenden zusätzlich *Reinforcement Learning from Human Feedback (RLHF)*, um die Modelle besser an menschliche Präferenzen anzupassen [45, 46].

2.3.2 State-of-the-Art LLM-Familien und Googles Gemini AI

Unter den aktuellen Marktführern befinden sich eine Handvoll State-of-the-Art-Modelle, deren Ergebnisse beachtlich sind. Dazu zählt neben OpenAI's ChatGPT die Claude-Familie von Anthropic sowie xAI's Grok-Varianten. Aufgrund der Wahl von Gemini für den praktischen Teil, wie in Kapitel 4 später genauer erläutert wird, ist dessen Aufbau und Funktionsweise besonders interessant.
Google DeepMind stellt seit wenigen Jahren [47] verschiedene Modelle der sogenannten Gemini-Familie zur Verfügung. Als Nachfolger von LaMDA (Language Model for Dialogue Applications) und PaLM 2(Pathways Language Model) wurden diese vor allem für die nahtlose Integration multimodaler Fähigkeiten entwickelt [44, 48, 49]. Im Gegensatz zu rein textbasierten Modellen wird hier die Heterogenität der Datenquellen in den Vordergrund gelegt, um so besser Zusammenhänge zwischen Text, Code, Bildern, Audio und Video zu verknüpfen. Die Synthese von multimodaler Information wird damit erfolgreich adressiert.
Den Kern bildet dabei die Sparse Mixture-of-Experts (MoE)-Architektur [50, 51, 52], die im Grunde auf der zuvor erwähnten Transformer-Architektur aufbaut. Im dynamischen Routing werden nur ein Teil der Expertengruppen pro Eingabetoken aktiviert, was im Kontrast zur normalen Herangehensweise steht, bei der üblicherweise alle Parameter aktiviert werden. Der Vorteil besteht hier nun vor allem darin, dass die Modellkapazität und der Rechenaufwand nahezu entkoppelt werden, sodass deutlich höhere Parameteranzahlen verwaltet werden können, was sich positiv auf die Inferenzzeit und die Hardwareanforderungen auswirkt [53, 54].

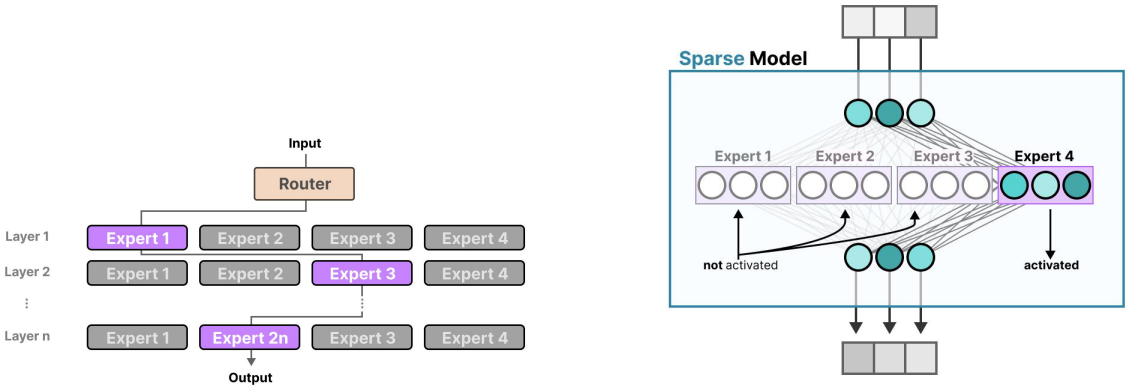


Figure 2.3: *Inklusion von Experten [55]*

Das in mehreren Stufen erfolgende Training beginnt mit dem Pre-Training, bei dem große Datenmengen aus Web-Dokumenten, Quellcodes, Bild- und Audiomaterial verwendet werden. Im Instruction-Tuning werden mittels spezieller Datensätze, die aus Aufgabenpaaren und menschlichen Präferenzdaten bestehen, nutzerspezifische Interaktionen geprägt. Der letzte Feinschliff erfolgt im letzten Schritt. Reinforcement Learning from Human Feedback verbessert die Qualität der Ausgaben durch Anpassung an menschliche Erwartungen [49].

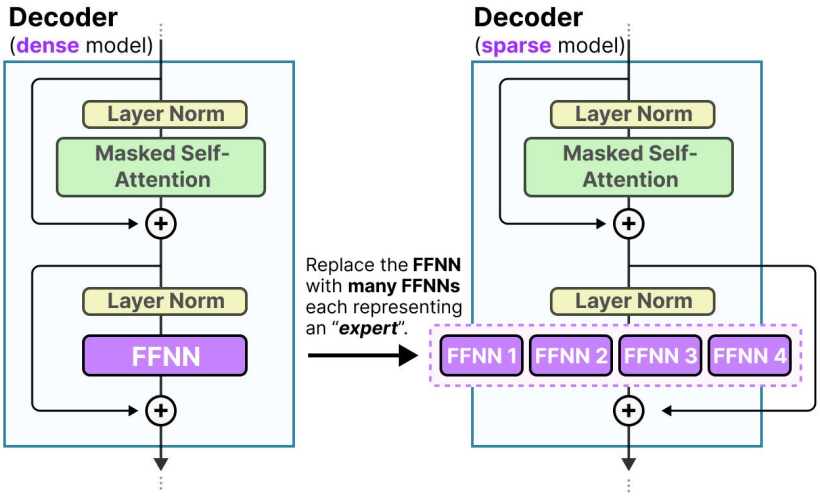


Figure 2.4: *Der Vergleich herkömmlicher und Expert integrierter Transformer [55]*

Google's intensiver Aufbau über die vergangenen Jahre in Tensor Processing Units (TPUs) erweist sich immer mehr als lukratives Investment [56, 57]. Durch verbesserte Parallelisierung und effizientere Trainingsabläufe können dabei in kürzerer Zeit größere Skalen erreicht werden. Dadurch wird auch die zu Beginn erwähnte native Multimodalität ermöglicht, die sich im Kontrast zu den adaptiven Systemen befindet, bei denen erst durch Adapter andere Formen von Daten verarbeitet werden können. Mit einem Kontext-Fenster von bis zu 1 Million Tokens und der adaptiven Fähigkeit, sowohl kreativ als auch logisch zu arbeiten, erweist sich Gemini als herausragende Basis.
Für das vorliegende Projekt liegt der Fokus neben nativer Interaktion und Unterhaltung auf Latenz und Performance. Während Varianten wie Gemini 2.5 Pro [58] durch tiefere Antworten beeindruckendere Ergebnisse bei komplexen Sachverhalten erzielen, ist die zeitliche Latenz nicht zu vernachlässigen. Dies schafft das "2.5 Flash LiteModell von Google [59, 60]. Mit der Verringerung von Experten werden die Parameter kompakter organisiert und sorgen für eine geringere Hardware-Last. Im Gegensatz zum verbreiterten Routing wird eine maximale Anzahl im sogenannten Top-K Routing pro Token angewandt.

2.3.3 API

Auch wenn es unter den verschiedenen Variationen vieler Large Language Modelle kleinere Versionen gibt, ist die Ausführung auf herkömmlicher Hardware weiterhin sehr rechenintensiv und teilweise nahezu unmöglich. Mit Nvidias Grafikkarten und CUDA-Systemen werden Rechenoperationen erst ermöglicht und schließen alle anderen nahezu vollständig aus. Um

diesen Hardware-Limitationen entgegenzukommen, wird für den Gebrauch von Gemini 2.5 Flash-Lite Google's offizielle API angewendet [61]. Mit einem Client-Request gelangen Informationen über Modell, Input und Parameter authentifiziert an die offiziellen Server. Mit einer Reduzierung der eigenen Last und der rechenoptimierten Auslastung der Cloud-Server wird die Antwort auf den internen TPUs optimiert generiert und im JSON-Format zurückgesendet, das sich im eigenen Programm wiederfinden lässt [62].
Wie beobachtet werden konnte, ist für die reibungslose Implementierung eine Vielzahl an technischen Komponenten nötig. Dies wird einerseits durch bestehende Forschungsarbeiten an Large Language Modellen und andererseits durch die zur Verfügung gestellte Hardware von Google ermöglicht. Aufgrund dieser Schritte werden sowohl Quantität, Qualität als auch Latenz sichergestellt [53]. Die Zukunft der LLM-Entwicklung verspricht weitere Fortschritte in Bereichen wie Multimodalität, Effizienz und Spezialisierung, während gleichzeitig Herausforderungen wie Halluzinationen, Bias und Skalierbarkeit angegangen werden müssen. Für Entwickler und Forscher ist es entscheidend, diese Grundlagen zu verstehen, um die Möglichkeiten dieser transformativen Technologie voll auszuschöpfen.

2.4 Grundlagen der Sprachverarbeitung: Speech-to-Text und Text-to-Speech

In modernen Mensch-Maschine-Interaktionen spielen Sprachverarbeitungstechnologien eine zentrale Rolle. Insbesondere die Umwandlung von gesprochener Sprache in Text (Speech-to-Text, STT) und von Text in gesprochene Sprache (Text-to-Speech, TTS) ist essenziell für intuitive und zugängliche Schnittstellen. Diese Technologien haben sich in den letzten Jahren rasant entwickelt, primär angetrieben durch Fortschritte im Bereich des maschinellen Lernens und tiefer neuronaler Netze.

2.4.1 Speech-to-Text (STT)

Speech-to-Text ist zuständig für die Transformation gesprochener Sprache in Text, bekannt als automatische Spracherkennung (ASR). Dadurch können Rechner und Systeme Sprache identifizieren und umwandeln, was die Effizienz der Kommunikation erhöht und die Barrierefreiheit verbessert [63, 64].
Der Prozess gliedert sich in verschiedene Schritte. Das analoge Sprachsignal wird zu Beginn über ein Mikrofon aufgenommen, digitalisiert und in kleinere Einheiten aufgeteilt. Die Grundlage bilden akustische Signale, die als Schallwellen vorliegen. Deren computergestützte Auswertung basiert primär auf Verfahren des Machine Learnings sowie der Computerlinguistik. Ein wesentlicher Teilschritt ist hierbei die Segmentierung des Audiomaterials in Phoneme – jene minimalen lautlichen Einheiten, die eine differenzierende Bedeutung tragen. Mithilfe von Algorithmen werden in diesen Lautfragmenten spezifische Muster detektiert und entsprechenden Vokabeln oder Phrasen zugeordnet. Leistungsstarke neuronale Netzwerkarchitekturen, insbesondere Transformer-Modelle, ermöglichen es dabei, komplexe Abhängigkeiten zwischen den Lauten zu modellieren und so kohärente Sätze zu synthetisieren [63]. Herausforderungen ergeben sich dabei gerade durch Akzente, Dialekte, unterschiedliche Geschwindigkeiten oder auch Hintergrundgeräusche.

VOSK

VOSK dient als Offline-Toolchain für automatische Spracherkennung und bietet eine Vielzahl von Modellen, deren Fokus auf der Transkription liegt. Die Architektur beruht auf den Kaldi-Komponenten, die sich vor allem durch Chain-Modelle, TDNN-F-Schichten (Factorized Time-Delay Neural Networks) und WFST-Decoder beweisen [65, 66].

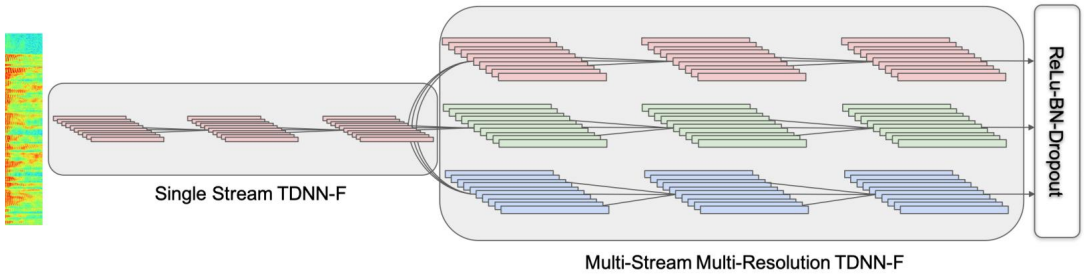


Figure 2.5: *Prinzip und Aufbau der VOSK-Architektur*

Die Basis bildet das Time Delay Neural Network (TDNN), das speziell für Sequenzen von Audio und Text entwickelt wurde. Um den Kontext und die Platzierung von Buchstaben einordnen zu können, betrachtet das TDNN nicht nur den aktuellen Moment, sondern auch die Fenster davor und danach. Erst dadurch wird Bedeutung und Symbolik korrekt eingeordnet [67, 68]. Aufgrund der hohen benötigten Rechenleistung und des Speichers muss eine Lösung gefunden werden, um die Kompatibilität mit schwacher Hardware wie Smartphones zu ermöglichen. Daher kam es zum Schritt in Richtung TDNN-F (Factored TDNN) [68, 69, 70]. Anstelle einer großen Berechnung werden zwei oder mehrere kleinere hintereinander ausgeführt. Dies führt unter geringerem Rechenaufwand und in kürzerer Zeit erstaunlicherweise zu den gleichen Ergebnissen. Der Workflow sieht dementsprechend folgendermaßen aus [67, 71]: Das Audiosignal gelangt an und wird in drei parallelen Streams verarbeitet. Dabei werden in jedem Stream mittels Subsampling die Audiosignale verdünnt. Die Streams zerlegen das Signal dementsprechend in Details, Silben und Rhythmen. Über Details hinweg wird also unter anderem nur alle paar Frames ein Blick auf den Input geworfen, sodass Längen, Fluss von Wörtern und Sprachrhythmus miteinander in Einklang gebracht werden. Jeder dieser Streams wird anschließend durch eine eigene TDNN-F-Schicht geschickt. Mit den drei Ergebnissen [17] wird eine finale Version zusammengestellt, in der grobe, mittlere und feine Details zusammengefügt wurden.

Faster Whisper

Das von OpenAI entwickelte Modell Whisper [72] ist ebenfalls dafür zuständig, Daten im Audioformat in Text umzuwandeln. Den Kern bildet eine Encoder-Decoder-Transformer-Architektur, die in zwei Hauptphasen unterteilt ist [73, 74].
Für den ersten Schritt wird das Audiosignal in für Maschinen verständliche Sprache umgewandelt. Dies spiegelt sich im Log-Mel-Spektrogramm wider. [75]. Durch die Visualisierung der Frequenzen erhält ein neuronales Netzwerk Zugriff und extrahiert die wesentlichen Merk-

male der gesprochenen Sprache [76]. In Form von numerischen Repräsentationen wird das Ergebnis geliefert, das den Inhalt und den Kontext der erfassten Inhalte enthält. Nach der ersten Verständnisphase folgt die Schreibphase. In dieser befindet sich der Decoder, der die numerischen Repräsentationen empfängt. Das dahinterstehende zweite neuronale Netzwerk ist nun für die Umwandlung in verständliche Textsequenzen zuständig. Wie auch bei den Large Language Models sind die Attention-Mechanismen ein fundamentaler Bestandteil, wenn es darum geht, die relevanten Teile zu extrahieren. [77, 78]. Das ist besonders dann wichtig, wenn über längeren Text hinweg der Kontext bewahrt werden soll. Mit über 680.000 Stunden [79, 80] Audiomaterial wurde das Modell trainiert und auf präzise Ergebnisse abgestimmt. Zur Verfügbarkeit zählen unterschiedliche Modellgrößen, die sich sowohl in Ressourcen als auch in Geschwindigkeit unterscheiden und je nach Zweck Vorteile liefern [81, 82, 83].

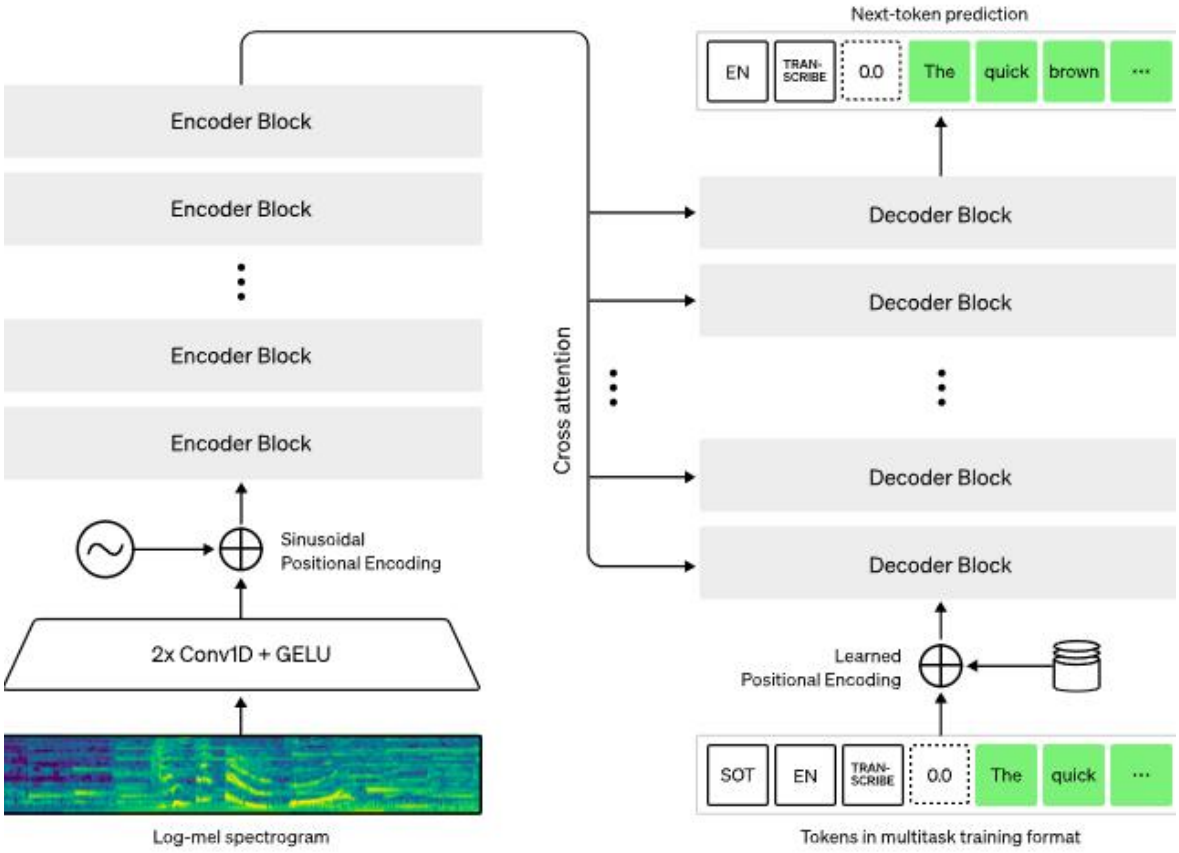


Figure 2.6: *Architektur des Basis Whisper Modells [72]*

Trotz seiner Leistungsfähigkeit ist das Modell in gewissen Anwendungen noch nicht leistungsstark genug und sorgt unter anderem bei der Echtzeittranskription nicht für die gewünschte Performance. Aus diesem Bedarf heraus hat sich das Faster-Whisper-Modell entwickelt. Dabei handelt es sich nicht um eine Neuentwicklung der Architektur [84], sondern eher um eine Reimplementierung des Whisper-Modells mittels C2Translate.[84, 85], welche

sich durch eine eigene Inferenz-Engine [86] für Transformer-Modelle in der zu beobachtenden Geschwindigkeit merklich macht. Die dabei verwendete Technik ermöglicht bis zu viermal schnellere Ergebnisse, indem die Präzision der mathematischen Operationen reduziert wird. Damit wird sowohl weniger Speicher belegt als auch schneller gerechnet, ohne signifikante Einbußen in der Genauigkeit. Darüber hinaus werden Rechenoperationen zusammengefasst und in der Reihenfolge variiert, um die Recheneinheiten wie CPU und GPU optimal auszulasten. Dank dieser Anpassungen kann Faster Whisper unter gleichbleibender Genauigkeit und geringerem Bedarf an Speicherkapazität die gleichen Ergebnisse auf schwächerer Hardware reproduzieren [79, 87].

Im Gegensatz zu Faster-Whisper handelt es sich bei der Whisper-Turbo-Variante um eine direkte Modifikation der Modellarchitektur. [79, 88]. Mit einer drastischen Reduzierung der Decoder-Schichten von 32 auf nur noch 4 ist die Inferenz wesentlich schneller [89, 90, 91, 92].

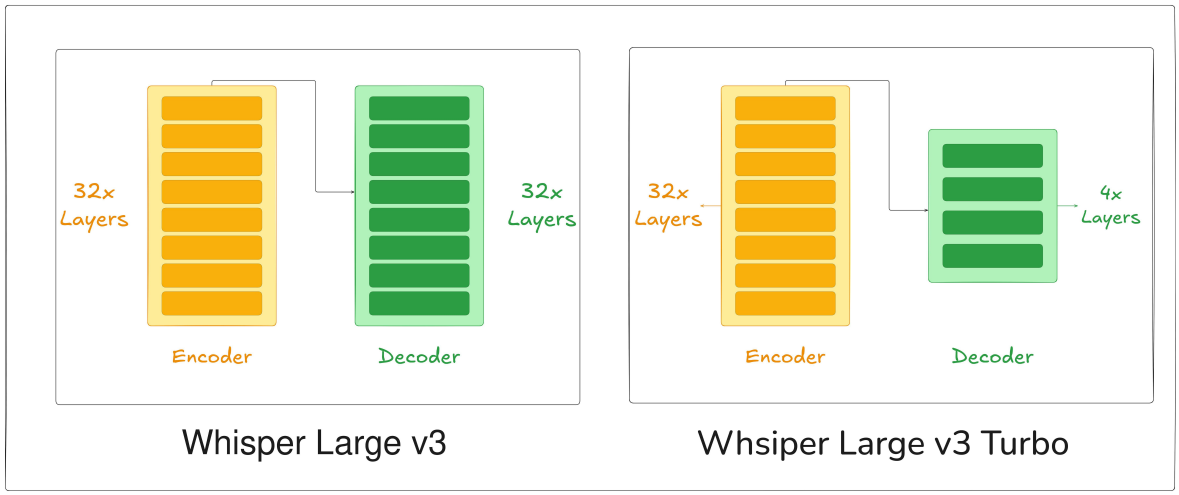


Figure 2.7: *Direkter Vergleich zwischen Basis und Turbo Variante des Whisper Modells mit Fokus auf die Anzahl der Decoder-Schichten [79]*

Inwiefern die Nutzung und Auswahl maßgeblich sind, wird in späteren Kapiteln genauer aufgezeigt.

2.4.2 Text-to-Speech (TTS)

Die Technik Text-to-Speech (TTS), auch als Sprachsynthese bekannt, kann geschriebenen Text in Sprache umwandeln und ermöglicht die künstliche Erzeugung einer menschlichen Stimme [93, 94, 95, 96]. TTS-Systeme werden oft in Kombination mit Spracherkennung eingesetzt, um die Interaktion mit Kommunikationssystemen zu erleichtern [93].
Zwei Hauptkomponenten prägen dabei den Aufbau eines solchen Systems. Die NLP-Komponente analysiert den Eingabetext, um ihn in phonetische Repräsentationen umzuwandeln. Dazu gehört die Segmentierung der Zeichenkette, die Bestimmung der Aussprache von Wörtern (Lexikondatenbank, Regeln für ungewöhnliche Wörter), die Analyse der Satzstruktur und die Vorhersage der Prosodie (Intonation, Rhythmus, Betonung, Pausen). [93, 94]. Im zweiten

Teil befindet sich das Digital Signal Processing (DSP). Basierend auf der phonetischen und prosodischen Repräsentation generiert diese Komponente das akustische Sprachsignal. Früher wurden oft analytische Methoden verwendet, während heute überwiegend tiefe neuronale Netze (Deep Learning Speech Synthesis) eingesetzt werden, die auf großen Mengen hochwertiger Sprachaufnahmen trainiert wurden, um natürlich klingende Stimmen zu erzeugen [95, 96, 97]. Techniken wie WaveNet und Tacotron von Google haben hier maßgebliche Fortschritte erzielt [94, 98, 99].

Tacotron 2

Tacotrons Fähigkeit, Text zu Sprache umzuwandeln, besteht aus zwei wesentlichen Bestandteilen [100]. Darunter zählt einerseits der Encoder, der den aufgenommenen Text in numerische Repräsentationen umwandelt [101]. Damit werden die phonetischen Eigenschaften der Wörter erfasst. Mittels Aufmerksamkeitsmechanismus wird erneut festgelegt, welche Teile von Bedeutung sind und später fokussiert betrachtet werden sollen. Im Gegensatz zu STT verläuft der Flow nun andersherum, was den Decoder an den Schluss stellt und im letzten Schritt das Mel-Spektrogramm erzeugen lässt [102]. Bei langen Texten treten jedoch häufiger Fehler auf, was auf den Aufmerksamkeitsmechanismus zurückzuführen ist. Die Lösung bietet die Double Decoder Consistency (DDCS) [100, 101]. Dadurch wird dem Modell ein zweiter Decoder zur Verfügung gestellt, der sich in der Feinheit der Analyse unterscheidet. Während ein Decoder ein sehr feines Mel-Spektrogramm erzeugt, arbeitet der zweite parallel in größeren Schritten. Mit einer geringeren Audioqualität werden somit auch weniger Aufmerksamkeitsfehler gemacht. Nach separaten Trainings werden beide Ergebnisse verglichen und angepasst, was am Ende zu deutlich verbesserten Sprachausgaben führt.

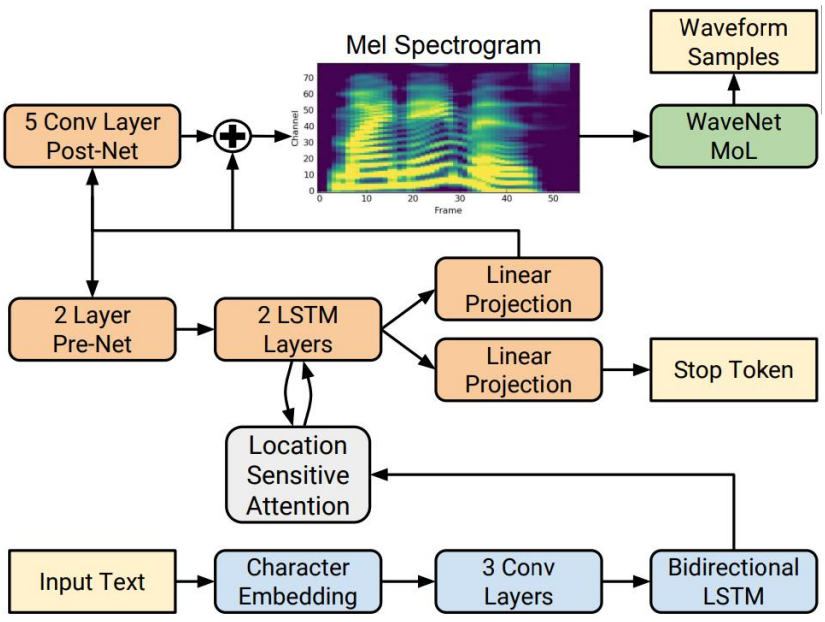


Figure 2.8: *Architektur und Komponenten von Tacotron 2 DDCS [100]*

Sesame CSM

Wenn es darum geht, die Sprache realistisch und natürlich klingen zu lassen, führt kein Weg an Sesame CSM 1b vorbei. Das Conversational Speech Model ist besonders gut darin, den Kontext des Gesprächs zu erfassen und sich dementsprechend anzupassen. Auch hier kommt eine leistungsstarke Transformer-Architektur zum Einsatz, die eine Abwandlung von Metas Llama-Modell ist [103, 104]. Hier werden semantische und kontextuelle Informationen gewonnen, die für ein ganzheitliches Verständnis wichtig sind. Dahinter ist ein spezieller Decoder gekoppelt, der daraufhin die Informationen in Audiosignale umwandelt. Speziell kommen hier Residual Vector Quantization (RVQ) und Mimi-Audio-Codes zum Einsatz. Diese Codes sind eine effiziente, komprimierte Darstellung der Sprachmelodie, des Rhythmus und der Klangfarbe. Zuletzt werden die Audio-Codes von einem Vocoder in hörbare Schallwellen umgewandelt.

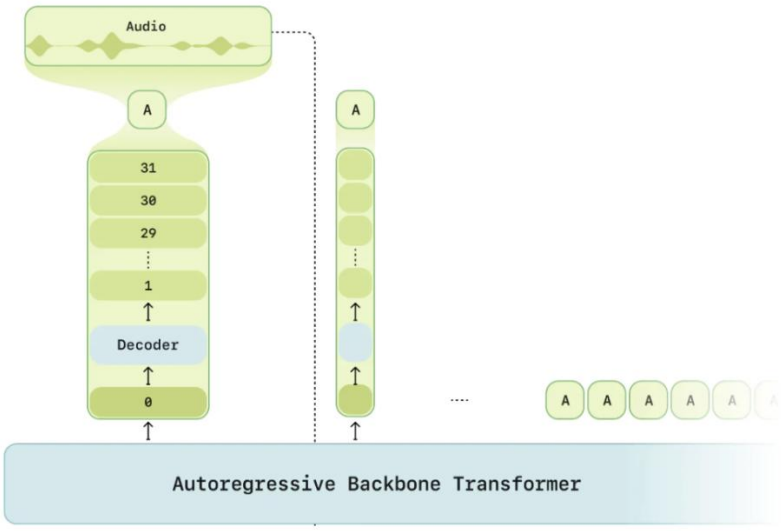


Figure 2.9: *Architektur und Komponenten von Sesame CSM 1b [105]*

Die Besonderheit liegt zum einen in der Fähigkeit, Gesprächsverläufe [104] zu berücksichtigen und Tonfall, Emotion oder auch Betonung an die Historie anzupassen, was sich besonders für natürliche Konversationen anbietet. Dies wird vor allem durch die Kombination mit starken Large Language Modellen ermöglicht, die auch hier zum Einsatz kommen. Darüber hinaus wird mittels der Mimi-Audio-Codec-Technologie rechenressourcensparend gearbeitet.

Fast Pitch

Das von Nvidia entwickelte FastPitch-Modell orientiert sich ebenfalls an einem Mel-Spektrogramm-Generator und einem Vocoder, der für die Umwandlung des Audiosignals in hörbare Audiodateien zuständig ist. [106, 107, 108]. Sein Vorteil in Bezug auf die Geschwindigkeit gegenüber Modellen wie Tacotron 2 wird durch die vollständig parallele Transformer-Architektur erreicht. Ältere Modelle arbeiten sequenziell und haben den Nachteil, dass Audio-Abschnitte nacheinander verarbeitet werden, was zu einer deutlich niedrigeren Geschwindigkeit führt. Dies umgeht FastPitch, indem der gesamte erhaltene Text auf einmal verarbeitet wird.[106].

Die Attention-Mechanismen schaffen es, die Synthese um ein Vielfaches schneller zu durchlaufen.

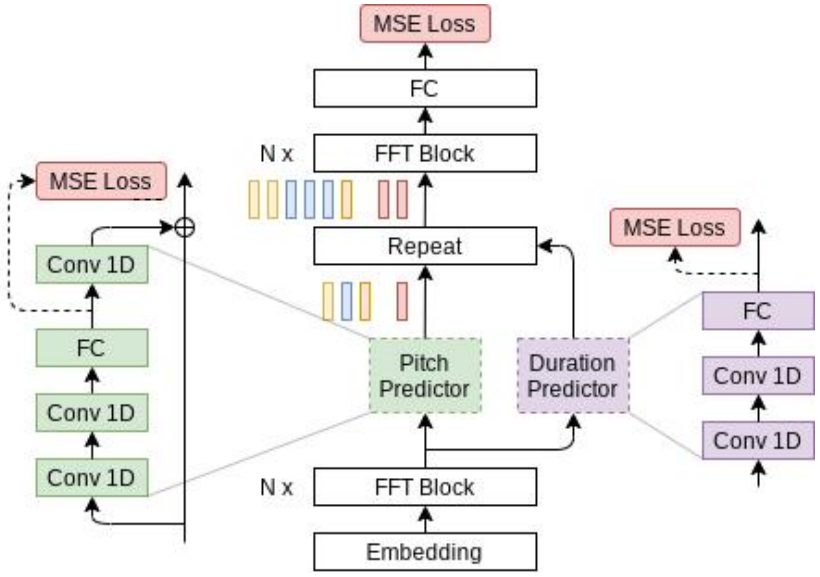


Figure 2.10: *Architektur und Komponenten von FastPitch [107]*

Ein entscheidender Aspekt der Innovation liegt in der Vorhersage und Kontrolle der Tonhöhe. Zuerst analysiert das Modell den Text und sagt voraus, wie lange jeder Buchstabe oder jedes Phonem (Sprachlaut) ausgesprochen werden muss. Anschließend sagt FastPitch für jeden dieser Laute den Verlauf der Tonhöhe voraus. Es erstellt also eine detaillierte Tonhöhenkurve für den gesamten Satz. Das Modell "konditioniert"dann die Erzeugung des Mel-Spektrogramms auf Basis dieser vorhergesagten Tonhöhenkurve. Das bedeutet, dass die Tonhöheninformationen direkt in die Erstellung des "Klang-Bauplansëinfließen. Dieser Fokus auf die Tonhöhe ist ein großer Vorteil. Er verbessert nicht nur die allgemeine Klangqualität der Sprache, sondern ermöglicht auch eine präzise Steuerung: Durch manuelle oder automatische Anpassung der vorhergesagten Tonhöhenkurve kann die erzeugte Sprache expressiver gestaltet werden, ganz ähnlich wie bei Sesame CSM

ESPnet2

ESPnet2 ist kein einzelnes Modell, sondern ein umfassendes, quelloffenes Toolkit für die Sprachverarbeitung [84, 109]. Es dient als Baukasten mit einer Vielzahl an Möglichkeiten zur Sprachverarbeitung. Der Ënd-to-EndAnsatz ist zentral für ESPnet2. Anstatt eine komplexe Aufgabe wie die Spracherkennung in viele kleine, separate Schritte (z. B. akustische Modellierung, Aussprachemodellierung, Sprachmodellierung) zu zerlegen, die einzeln trainiert werden, lernt ein einziges, tiefes neuronales Netzwerk, die gesamte Aufgabe von Anfang bis Ende zu bewältigen. Das Modell nimmt Text als Eingabe und erzeugt direkt die Audiodaten als Ausgabe [110, 111]. Die meisten Modelle, die mit ESPnet2 gebaut werden, folgen einer Encoder-Decoder-Architektur, die je nach Aufgabe leicht variiert. Im gleichen Sinne wie zuvor kommen Encoder und Decoder zum Einsatz, die für Aufnahme und Verarbeitung zuständig sind. Ein wesentliches Merkmal von ESPnet2 ist seine Flexibilität und Effizienz, die durch

zwei Neuerungen im Vergleich zu seinem Vorgänger erreicht werden [112, 113]. ESPnet2 ist nicht mehr zwingend auf das traditionelle Kaldi-Toolkit für die Sprachverarbeitung angewiesen. Das macht die Installation und Nutzung deutlich einfacher. Anstatt alle Audio- und Textdaten vor dem Training in ein spezielles Format umwandeln und speichern zu müssen, kann ESPnet2 dies ön the fly", also direkt während des Trainingsprozesses, erledigen. Es kann rohe Audiodaten direkt verarbeiten, was Speicherplatz spart und den Vorbereitungsaufwand erheblich reduziert [112, 113].

Pyttsx3

pyttsx3 ist eine plattformübergreifende Abstraktionsschicht, die keine eigene Sprachsynthese durchführt, sondern als Wrapper für die nativen TTS-Engines des jeweiligen Betriebssystems (z. B. SAPI5, eSpeak) dient [114, 115]. Im fundamentalen Gegensatz zu allen zuvor genannten neuronalen Ansätzen verzichtet es auf ein eigenes KI-Modell, was zu minimalem Ressourcenbedarf und einfacher Implementierung führt, jedoch die Sprachqualität und Flexibilität stark limitiert [116, 117].

2.5 Grundlagen der automatisierten Gesichtsverarbeitung

Die Gesichtserkennung (Face Detection) ermöglicht es, Gesichter in Videos oder Fotos zu identifizieren und zu lokalisieren. [118, 119]. Sie ist der erste und grundlegende Schritt in vielen Anwendungen der Computer Vision, insbesondere vor der Gesichtserkennung (Face Recognition), bei der die identifizierte Person erkannt wird [120, 121]. Schlüsselbegriffe in diesem Bereich sind Computer Vision, Objekterkennung und Gesichtslokalisierung.

Die Effizienz der Gesichtserkennung ist von entscheidender Bedeutung, insbesondere für den Einsatz auf Low-Power-Hardware wie eingebetteten Systemen und Mobilgeräten. Diese Systeme haben oft begrenzte Rechenleistung, Speicher und Batterielaufzeit. Anwendungen wie Echtzeit-Videoanalyse, batteriebetriebene Geräte, Augmented Reality und Edge Computing treiben den Bedarf an hoher Recheneffizienz und geringem Stromverbrauch voran [122, 123]. Die Pipeline besteht im Grunde aus der Lokalisierung des Gesichts, gefolgt von der geometrischen Normalisierung. Mittels korrigierter Ausrichtung von Merkmalen wie Augen werden die nachfolgend durchgeführten Schritte robuster. Folgend geschehen die für die Identifikation maßgeblichen Unterschritte. Aus dem ausgerichteten Gesichtsbild wird ein kompakter und diskriminativer Merkmalsvektor, ein sogenanntes Embedding, extrahiert. Dieser Vektor repräsentiert die einzigartigen Charakteristika des Gesichts in einem hochdimensionalen Raum. Der extrahierte Merkmalsvektor wird mit einem oder mehreren Vektoren in einer Datenbank verglichen, um basierend auf einer Distanz- oder Ähnlichkeitsmetrik eine Entscheidung über die Identität zu treffen. Mit einem Blick auf die Variabilität ergeben sich deutliche Herausforderungen. Neben Unterschieden in Gesichtsformen spielen Faktoren wie Pose, Beleuchtungsdifferenzen und Ausdrücke eine Rolle. Genauso sind Altersunterschiede und partielle Verdeckungen von Bedeutung. Die Leistungsfähigkeit der in dieser Arbeit verwendeten Modelle muss stets im Kontext dieser Herausforderungen bewertet werden.

2.5.1 Verfahren zur Gesichtsdetektion

Die im ersten Schritt notwendige und zuverlässige Detektion von Gesichtern ist erforderlich, um anschließend Gesichter zuordnen zu können. Die in dieser Arbeit verglichenen Modelle teilen sich in zwei Ansätze auf, die hier näher erläutert werden.

Klassische Machine-Learning-Ansätze

Die Gesichtsdetektion basierte ursprünglich auf manuell definierten Merkmalen. Dies trug dazu bei, viele Eigenschaften von Gesichtern zu modellieren. Aufgrund der hohen Effizienz und Interpretierbarkeit war dies über viele Jahre der Standard, bevor neuronale Netzwerke übernahmen.

Haar Cascade Classifier verwenden rechteckige Kontrastmuster [124]. Mit einer Kodierung grundlegender visueller Strukturen werden typische Merkmale charakterisiert. Der Wert berechnet sich aus der Differenz der Summe der Pixelintensitäten der schwarzen und weißen Rechtecke. [125**?**]. Die Berechnung von tausenden Haar-Merkmalen in allen Positionen und Skalierungen ist jedoch zeitaufwendig. Eine Beschleunigung wird durch eine intermediäre Bildrepräsentation namens Integralbild erreicht. [126]. Hier entspricht jeder Wert jedes Pixels der Summe aller Pixelintensitäten innerhalb jedes beliebigen Rechtecks. Aus einem riesigen Pool an potenziellen Merkmalen wählt Adaptive Boosting eine kleine Teilmenge der diskriminativsten Merkmale aus. AdaBoost trainiert iterativ eine Reihe von Klassifikatoren, die nur auf einem einzigen Merkmal basieren. Fälschlich klassifizierte Merkmale erhalten mehr Gewicht, sodass in weiteren Durchläufen der Fokus auf diesen Bereichen liegt. Das Ergebnis spiegelt einen starken Klassifikator wider [127]. Die tatsächliche Effizienz des Detektors wird durch die Anordnung der ausgewählten Merkmale in einer Kaskade von Klassifikatoren erreicht. Diese Kaskade besteht aus mehreren Stufen, wobei die ersten Stufen nur wenige, aber sehr effektive Merkmale enthalten. Ein zu prüfender Bildausschnitt wird nur dann zur nächsten Stufe weitergeleitet, wenn er alle vorherigen Stufen erfolgreich passiert. Da die meisten Bildbereiche kein Gesicht enthalten, können sie bereits in den ersten, rechnerisch günstigen Stufen schnell verworfen werden. Dies fokussiert die Rechenleistung auf vielversprechende Kandidaten.

Eine Weiterentwicklung der kaskadierenden Detektoren stellt die Verwendung von Local Binary Patterns (LBP) anstelle von Haar-Merkmalen dar. LBP ist ein robuster Texturdeskriptor, der die lokale räumliche Struktur eines Bildes beschreibt [128, 129]. Der klassische LBP-Operator funktioniert, indem er den Grauwert jedes Pixels mit den Grauwerten seiner acht direkten Nachbarn vergleicht. Ist der Wert des Nachbarpixels größer oder gleich dem des Zentralpixels, wird eine 1 notiert, andernfalls eine 0. Das Ergebnis ist ein 8-Bit-Code, der die lokale Textur repräsentiert. Die Stärke von LBP-basierten Detektoren liegt in ihrer Robustheit gegenüber monotonen Helligkeitsänderungen sowie ihrer vergleichsweise geringeren Rechenkomplexität [130]. Während Haar-Merkmale mit Fließkommazahlen arbeiten, basieren LBP auf ganzzahligen Operationen, was sie insbesondere für den Einsatz in ressourcenbeschränkten Systemen vorteilhaft macht. In der Praxis werden LBP-Merkmale analog zu den Haar-Merkmalen in einer kaskadierten Struktur mit einem Boosting-Verfahren trainiert, um eine schnelle Detektion zu ermöglichen [131]. Im Vergleich zu Haar-Kaskaden sind LBP-Kaskaden

oft schneller, können jedoch eine etwas geringere Detektionsgenauigkeit aufweisen.

Der von Navneet Dalal und Bill Triggs im Jahr 2005 popularisierte Histogram of Oriented Gradients (HOG)-Deskriptor ist ein weiterer einflussreicher Ansatz zur Merkmalsextraktion für die Objekterkennung [132, 133]. Die grundlegende Idee ist, dass die Form und das Erscheinungsbild eines lokalen Objekts durch die Verteilung von Intensitätsgradienten oder Kantenausrichtungen gut charakterisiert werden können [134**?**]. Dabei sind vier Schritte von Bedeutung. Der Prozess zur Erstellung eines Histogramms orientierter Gradienten (HOG-Deskriptors) zur Merkmalsextraktion in Bildern beginnt mit der Berechnung der horizontalen und vertikalen Gradienten für jedes Pixel, um Kanteninformationen zu erfassen [129]. Anschließend wird das Bild in kleine Bereiche, sogenannte Zellen, unterteilt. Für jede dieser Zellen wird ein Histogramm erstellt, das die Verteilung der Gradientenrichtungen der darin enthaltenen Pixel darstellt [129]. Um den Deskriptor unempfindlicher gegenüber Änderungen in der Beleuchtung und im Kontrast zu machen, werden benachbarte Zellen zu größeren Blöcken zusammengefasst und die Histogramme innerhalb jedes Blocks gemeinsam normalisiert [135]. Der finale HOG-Merkmalsvektor, der für Aufgaben wie die Objekterkennung verwendet werden kann, entsteht schließlich durch das Zusammenfügen der normalisierten Histogramme aller Blöcke. In der Dlib-Bibliothek wird dieser HOG-Merkmalsvektor verwendet, um einen linearen Support-Vector-Machine (SVM) Klassifikator zu trainieren [132, 135]. Dieser Detektor wird dann in Kombination mit einer Bildpyramide und einem Sliding-Window-Ansatz auf das gesamte Bild angewendet, um Gesichter unterschiedlicher Größe zu lokalisieren [134]. HOG-Detektoren sind oft robuster gegenüber leichten Variationen in Pose und Beleuchtung als Haar-Kaskaden, haben jedoch tendenziell Schwierigkeiten mit stark rotierten oder nichtfrontalen Gesichtern.



Figure 2.11: *Prinzipieller Algorithmus bei Haar-Kaskaden [136] und Dlib Hog [134]*

Deep-Learning-basierte Ansätze

Mit dem Aufstieg des Deep Learning, insbesondere der Convolutional Neural Networks (CNNs), vollzog sich ein fundamentaler Paradigmenwechsel in der Objektdetektion. Im Gegensatz zu den klassischen Ansätzen, die auf manuell entworfenen Merkmalen basieren, lernen CNNs hierarchische Merkmalsrepräsentationen direkt aus den Rohdaten . Diese datengesteuerte Merkmalsextraktion führt zu einer signifikant höheren Genauigkeit und einer gesteigerten Robustheit gegenüber den genannten Herausforderungen, wie Variationen in Pose, Beleuchtung

und Mimik [137, 138, 139]. Die in dieser Arbeit verwendeten Modelle gehören überwiegend zur Familie der sogenannten Öne-StageDetektoren. Diese zeichnen sich dadurch aus, dass sie die Lokalisation (Bounding Boxes) und die Klassifizierung (Gesicht vs. Nicht-Gesicht) in einem einzigen Durchlauf durch das neuronale Netz durchführen, was sie besonders effizient macht.

Der Single Shot MultiBox Detector (SSD) ist eine weit verbreitete und effiziente Architektur für die Objekterkennung [140]. Das Grundprinzip des SSD-Frameworks besteht darin, ein Basis-CNN (oft ein vortrainiertes Klassifikationsnetzwerk wie ResNet) als Merkmalsextraktor zu verwenden. Die Innovation von SSD liegt in der Nutzung von Merkmalskarten (feature maps) aus mehreren, über das Netzwerk verteilten Konvolutionsebenen. Frühere Ebenen mit hoher räumlicher Auflösung sind für die Detektion kleiner Objekte zuständig, während spätere Ebenen mit geringerer Auflösung, aber reicherem semantischen Kontext größere Objekte erkennen. Für jede Position auf diesen Merkmalskarten werden Vorhersagen für eine Reihe von vordefinierten "Default Boxes"(auch Anchor Boxes genannt) unterschiedlicher Größen und Seitenverhältnisse getroffen. Das Netzwerk lernt, sowohl die Wahrscheinlichkeit für das Vorhandensein eines Gesichts als auch die notwendigen Anpassungen (Offsets) dieser Default Boxes vorherzusagen, um das Gesicht präzise zu umschließen. Die Implementierung im DNN-Modul von OpenCV ermöglicht eine zugängliche Nutzung dieses leistungsstarken Modells [141].

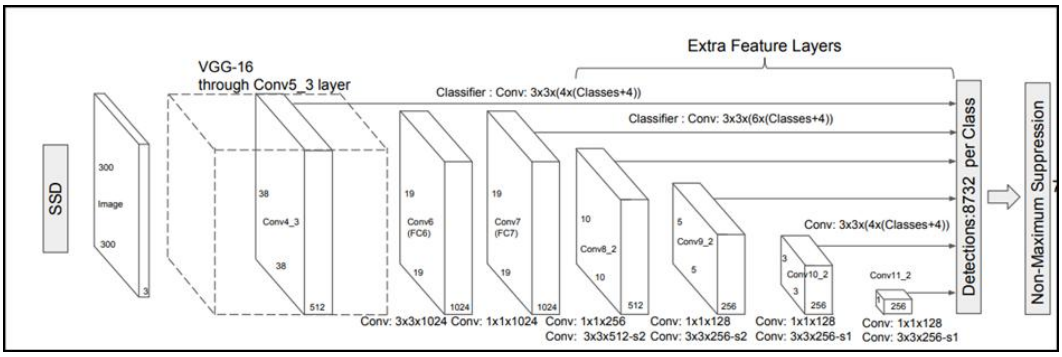


Figure 2.12: *Vielzahl an Schichten einer Deep Neural Network Single Shot Detection Architektur [142]*

YuNet ist ein leichtgewichtiger und hochperformanter CNN-basierter Gesichtsdetektor, der speziell für Echtzeitanwendungen auf CPUs optimiert wurde [143]. Das Modell stellt einen exzellenten Kompromiss zwischen Detektionsgenauigkeit und Inferenzgeschwindigkeit dar. Durch seine schlanke Architektur eignet es sich besonders für den Einsatz in ressourcenbeschränkten Umgebungen, in denen der Einsatz größerer Modelle wie SSD oder YOLO rechentechnisch nicht realisierbar wäre. YuNet ist in modernen Versionen der OpenCV-Bibliothek integriert und wird dort als performanter Nachfolger der klassischen Haar- und LBP-Kaskaden positioniert [144].

MediaPipe ist ein von Google entwickeltes, plattformübergreifendes Framework für den

Aufbau von multimodalen Wahrnehmungspipelines [145, 146]. Die darin enthaltene Gesichtsdetektionslösung ist für ihre extreme Geschwindigkeit und Effizienz bekannt, die den Echtzeiteinsatz auf mobilen Endgeräten und im Webbrowser ermöglicht. Das zugrunde liegende Modell, oft eine Variante von BlazeFace, wurde gezielt für diese Anwendungsfälle konzipiert. Eine Besonderheit ist, dass die MediaPipe-Lösung über die reine Bounding-Box-Prädiktion hinausgeht und in der Regel direkt sechs wesentliche Gesichtslandmarken (Augen, Nase, Mundwinkel) mitliefert [147, 148]. Diese zusätzlichen Informationen sind für nachfolgende Schritte wie die Gesichtsausrichtung von großem Wert und können die Notwendigkeit eines separaten Landmarken-Detektors eliminieren.

YOLO (You Only Look Once) ist eine der bekanntesten und einflussreichsten Familien von Öne-StageObjektdetektoren. Das Kernkonzept von YOLO besteht darin, das Eingangsbild in ein Raster zu unterteilen [149, 150]. Jede Zelle dieses Rasters ist dafür verantwortlich, Objekte zu detektieren, deren Mittelpunkt in sie fällt. YOLOv8 repräsentiert eine der neuesten und leistungsfähigsten Iterationen dieser Architektur und zeichnet sich durch eine hohe Flexibilität sowie eine herausragende Balance zwischen Geschwindigkeit und Genauigkeit aus. Das hier verwendete Modell, YOLOv8 Face, ist eine spezialisierte Version, die auf umfangreichen Gesichtsdatensätzen trainiert wurde. Es ist nicht nur in der Lage, Bounding Boxes zu prädizieren, sondern oft auch, ein Set von Gesichtslandmarken zu detektieren, was seine Nützlichkeit für eine komplette Gesichtsverarbeitungspipeline weiter erhöht [151].

Während ArcFace primär als ein hochmodernes Verfahren zur Gesichtserkennung bekannt ist, umfasst das als "Buffalo"bekannte Modellpaket eine vollständige Pipeline, zu der auch eine dedizierte Detektionskomponente gehört [152, 153]. Dieser Detektor ist darauf optimiert, Gesichter in einer Qualität zu lokalisieren und zuzuschneiden, die eine maximale Leistung des nachgeschalteten ArcFace-Erkennungsmodells gewährleistet. Technologisch basiert dieser Detektor häufig auf einer Architektur wie RetinaFace, die eine signifikante Verbesserung gegenüber dem Standard-SSD darstellt [153, 154]. Die Schlüssel-Innovation von RetinaFace ist die Einführung der "Focal LossFunktion während des Trainings. Diese Verlustfunktion reduziert den Einfluss von leicht klassifizierbaren Negativbeispielen (z. B. einfachen Hintergründen) und zwingt das Modell, sich stärker auf die schwierigen Fälle (z. B. kleine, verdeckte oder schlecht beleuchtete Gesichter) zu konzentrieren [123]. Das Ergebnis ist eine überlegene Detektionsleistung, insbesondere unter anspruchsvollen Bedingungen [154].

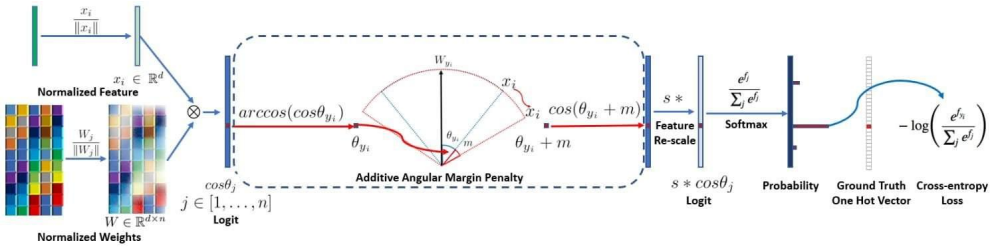


Figure 2.13: *Besonderer Fokus der additien Angular Margin Loss-Funktion im Arcface Aufbau [155]*

2.5.2 Verfahren zur Gesichtserkennung

Nach der erfolgreichen Detektion und gegebenenfalls Ausrichtung eines Gesichts folgt die eigentliche Erkennungsphase. Moderne Systeme der Gesichtserkennung basieren überwiegend auf Deep Learning. Anstatt vollständige Bildausschnitte direkt zu vergleichen, erzeugt ein tiefes neuronales Netz eine kompakte, numerische Repräsentation des Gesichts – ein sogenanntes Embedding.

Das Ziel ist es, die hochdimensionalen Bilddaten in einen niedrigdimensionalen Merkmalsraum abzubilden. In diesem Merkmalsraum gilt: Gesichter derselben Person sollen nahe beieinander liegen (intra-Klassen-Kompaktheit), während Gesichter verschiedener Personen möglichst weit voneinander entfernt sind (*inter-Klassen-Trennung*).

Die Identität einer Person wird anschließend nicht durch direkten Bildvergleich, sondern durch Distanzmetriken wie den euklidischen Abstand oder die Kosinus-Ähnlichkeit bestimmt. Besonders die Kosinus-Ähnlichkeit ist verbreitet, da sie robust gegenüber Variationen in der Bildbeleuchtung ist. Die Entscheidung über eine Übereinstimmung erfolgt durch einen Schwellenwertvergleich. Der Erfolg eines solchen Systems hängt maßgeblich davon ab, wie gut das neuronale Netz hochdifferenzierende Embeddings erzeugt, wobei sowohl die Architektur als auch die verwendete Verlustfunktion entscheidend sind.

OpenFace (basierend auf FaceNet)

OpenFace ist eine Open-Source-Implementierung, die auf der einflussreichen FaceNet-Architektur von Google basiert [156]. Die zentrale Innovation liegt in der Nutzung der *Triplet Loss Function*. Dabei werden drei Bilder betrachtet: ein Anker, ein positives Beispiel derselben Person und ein negatives Beispiel einer anderen Person. Ziel ist es, den Abstand zwischen Anker und Positiv zu minimieren, während der Abstand zwischen Anker und Negativ maximiert wird – mit einem definierten Sicherheitsabstand (*Margin*). Das Modell lernt somit direkt eine geeignete Einbettung anstelle einer klassischen Klassifikation.

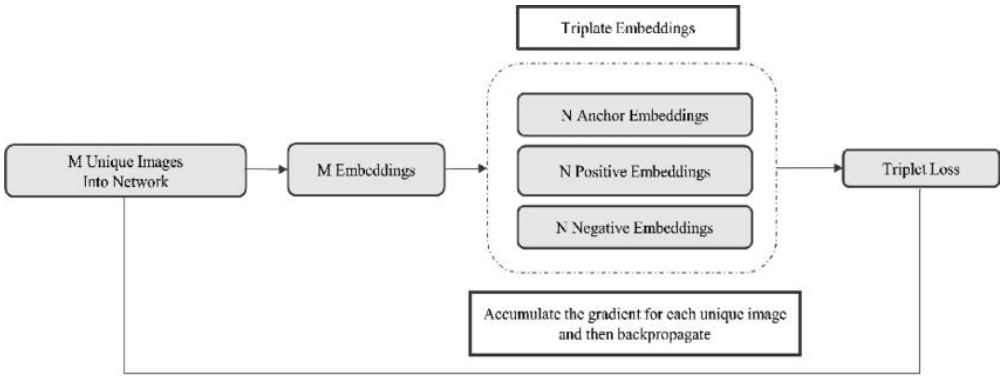


Figure 2.14: *Dreifacher Embedding Ansatz bei OpenFace [157]*

VGG-Face2

VGG-Face2 ist sowohl ein umfangreicher Gesichtsdatensatz als auch die darauf trainierte Modellfamilie [158]. Mit über 3,3 Millionen Bildern von mehr als 9.000 Personen deckt der

Datensatz eine große Bandbreite an Posen, Altersgruppen, Beleuchtung und Ethnizität ab. Trainiert wurden Architekturen wie ResNet-50 und SENet, die besonders robuste und generalisierbare Gesichtsrepräsentationen erzeugen. Die Stärke von VGG-Face2 liegt daher vor allem in der Datenvielfalt, die eine gute Generalisierung ermöglicht.

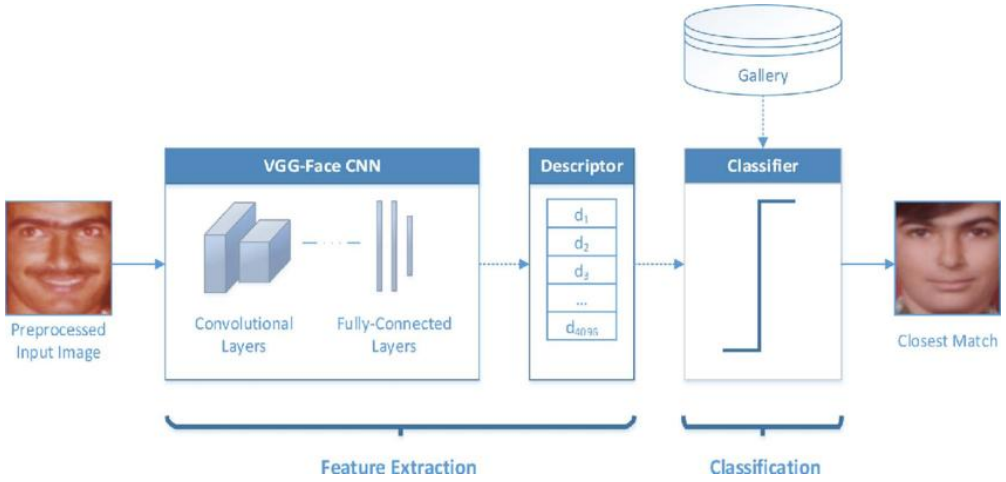


Figure 2.15: *Aufbau und Architektur von VGG2 [159]*

ArcFace (Buffalo)

ArcFace stellt eine wichtige Weiterentwicklung im Bereich der Verlustfunktionen dar [153]. Die Innovation liegt in der sogenannten Additive Angular Margin Loss, die im Winkelraum arbeitet. Durch die Normalisierung der Vektoren hängt die Vorhersage nur vom Winkel zwischen ihnen ab. Eine zusätzliche Winkelmarge erzwingt eine klarere Trennung der Identitäten und führt zu Embeddings mit hoher intra-Klassen-Kompaktheit und großer inter-Klassen-Distanz. Unter der Bezeichnung 'Buffalo' werden häufig optimierte Implementierungspakete verstanden, die ArcFace mit Detektions- und Alignment-Modellen kombinieren.

SFace

SFace wurde speziell für den Einsatz auf ressourcenbeschränkten Geräten wie Smartphones oder eingebetteten Systemen entwickelt [159]. Es basiert auf leichten CNN-Architekturen (z. B. MobileNet) und ist auf hohe Effizienz und geringe Latenz optimiert. Obwohl es nicht die Spitzenleistung großer Modelle wie ArcFace erreicht, bietet es eine gute Balance zwischen Genauigkeit und Geschwindigkeit und ist daher für Echtzeitanwendungen praxistauglich.

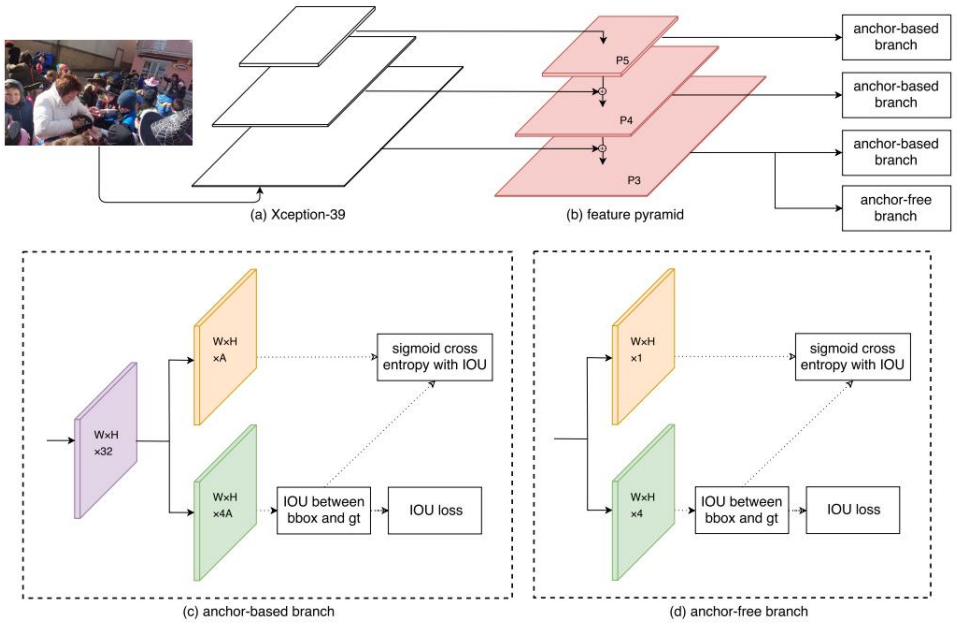


Figure 2.16: *Effizienzoptimierter Aufbau von SFace [159]*

2.5.3 Die Bedeutung vortrainierter Modelle

Heutige, leistungsstarke Deep-Learning-Modelle wie ResNet-50 oder YOLO verfügen über Millionen von Parametern. Der 'Start from Scratch'-Ansatz für das Training dieser Netzwerke ist deshalb extrem ressourcen- und kostenintensiv: Spezielle Hardware und Daten-Hubs sind zwingende Voraussetzungen für herausragende Computer, und einzelne Trainingsepisoden können mehrere Wochen dauern. Dazu kommt, dass diese Modelle auf gigantischen Datenmengen trainiert werden, die zusätzlich meist von Menschen speziell für die jeweilige Nische des Machine Learning vorbereitet werden – so kann natürlich kaum ein industrielles oder akademisches Computer-Vision-Projekt die Trainingsexperten dafür abstellen. Hier kommt auch Transfer Learning ins Spiel. Transfer Learning verwendet ein bereits trainiertes tiefes Netzwerk, das auf einer breiten Aufgabe trainiert wurde, als Basisnetzwerkstruktur und wird dann auf ein neues Problem spezialisiert. Die Motivation hinter dieser Methodik ist, dass die frühen Schichten eines tiefen Netzwerks allgemeine Muster aufnehmen, wie Kombinationen von Ecken, Linien und Texturen. Anstatt diese allgemeinen Merkmale neu zu lernen, werden die gleichen Gewichte der ersten Netze verwendet, jedoch mit modifiziertem Output. Es gibt jedoch entscheidende Vorteile: Der weitaus größte ist natürlich, dass es viel einfacher zu berechnen ist – der harte Teil des Lernens von Pixeln und des Überlappens von Kanten und Texturen wurde bereits von anderen abgeschlossen. Daher können wir hauptsächlich feintunen – genaue Parameter mithilfe spezifischer Trainingsdaten lernen. Zweitens sind 'tief trainierte' Modelle – Modelle, die auf riesigen, vielfältigen Datensätzen trainiert wurden – in der Regel viel robuster und gültiger als neu trainierte, sehr ähnliche Modelle. Dies ist ein weiterer wichtiger Punkt: Neue Modelle können nicht gut verallgemeinern. Da wir keine Dutzenden verschiedener Perspektiven für die Gesichtserkennung zur Verfügung haben und auch nicht

die Tausenden verschiedenen, einzigartigen Gesichter trainieren können, werden neu künstlich trainierte Modelle niemals so funktionsfähig sein wie tief trainierte State-of-the-Art-Modelle. Am Ende benötigen wir deutlich weniger Daten. Kurz gesagt: Wenn wir jemals ein tief trainiertes Gesichtserkennungsmodell bilden, benötigen wir nicht genügend Daten. Das ist ein großer Vorteil, obwohl er nicht so offensichtlich ist. Zusammenfassend nutzen alle im Rahmen dieser Arbeit verwendeten Modelle einige der neuesten Errungenschaften im Bereich des Transferschulens und basieren auf jahrelangem Wissen, das zuvor in riesige Trainingsmissionen eingebunden war. Ihre Vorfahren sind das Ergebnis von Tausenden bis Zehntausenden von Trainingsstunden – und ihre direkten Vorgänger umgehen diesen Prozess vollständig.

2.6 Egocentric Vision Systeme und KI-Assistenten

2.6.1 Einführung in die egozentrische Vision und Datenerfassung

Die egozentrische Vision ist ein Paradigma des maschinellen Sehens, bei dem Daten aus der Ich-Perspektive durch am Körper getragene Kameras und Sensoren erfasst werden. Dieser Ansatz simuliert die menschliche visuelle Erfahrung und liefert authentische Informationen für intelligente Anwendungen [160]. Die Forschung in diesem Bereich wird durch Fortschritte in der künstlichen Intelligenz und tragbaren Technologien vorangetrieben und findet praktische Anwendung in Augmented Reality, Virtual Reality und der Mensch-Maschine-Interaktion [160].

Meta hat mit der Entwicklung von Geräten wie den Meta ARIA-Brillen, die mit Kameras, Inertialmesseinheiten (IMUs) und GPS ausgestattet sind, maßgeblich zur Datenerfassung beigetragen [161]. Das Projekt Aria dient als Forschungsplattform, die egozentrische Daten nutzt, um die KI-Forschung voranzutreiben [162]. Die daraus resultierenden Erkenntnisse fließen in Verbraucherprodukte wie die Ray-Ban Meta Smart Glasses ein, die über eine hochauflösende Kamera und ein Fünf-Mikrofon-Array verfügen [163]. Der Datenschutz wird durch visuelle Indikatoren wie eine "Capture LED"berücksichtigt, die anzeigt, wenn das Gerät aufzeichnet [162, 163].

Die Entwicklung robuster KI-Systeme erfordert umfangreiche Datensätze. Ego4D ist mit über 3.670 Stunden Videomaterial von mehr als 923 Teilnehmern einer der größten egozentrischen Datensätze [164]. Er unterstützt Benchmarks in Bereichen wie episodisches Gedächtnis, Hand-Objekt-Interaktionen und soziales Verständnis [164]. Ein weiteres wichtiges Projekt ist EgoLife, das einen 300-stündigen Datensatz für die Entwicklung eines egozentrischen Lebensassistenten bereitstellt und auf ungestellten Momenten des täglichen Lebens basiert [165, 166].



Figure 2.17: *Ausschnitte aus dem Ego4D Datensatz mit Fokus auf Aufnahmen aus der Ego-Perspektive [167]*

2.6.2 Die Bedeutung qualitativ hochwertiger Datensätze

Trotz dieser Fortschritte bestehen Herausforderungen bei der Erfassung egozentrischer Daten. Dazu gehören die ständige Präsenz der Person im Video, die multimodale Natur der Daten und dynamische Perspektivwechsel durch Kopfbewegungen, die zu Verwacklungen und Verdeckungen führen [160]. Obwohl große Datensätze wie Ego4D existieren, gelten sie im Vergleich zu exozentrischen Datensätzen als unzureichend für das Training verschiedener Pre-Training-Modelle [160]. Die Erfassung von langen, kontinuierlichen egozentrischen Videos ist arbeitsintensiv und wirft erhebliche Datenschutzbedenken auf [168].

2.6.3 Forschung und Implementierungen von egozentrischen KI-Assistentendiensten

Tragbare KI-Technologien entwickeln sich von der reinen Datenerfassung hin zu proaktiver und personalisierter Unterstützung in Echtzeit. Insbesondere Smart Glasses eignen sich aufgrund ihres kompakten Designs für den kontinuierlichen täglichen Gebrauch und ermöglichen AR-Ansichten und freihändige Interaktionen [169, 170].

Ein Beispiel für ein fortschrittliches KI-Assistenzsystem ist Vinci, ein echtzeitfähiger, multimodaler egozentrischer Assistent für tragbare Geräte. Vinci arbeitet im Älways-onModus und nutzt das Modell EgoVideo-VL, um Szenen zu verstehen, vergangene Ereignisse abzufragen, Videos zusammenzufassen und zukünftige Aktionen zu planen [171, 172]. Das EgoLife-

Datensatz	Umfang (Std.) / Teilnehmer	Bereiche	Primärer Fokus/Zweck
Ego4D	~3.670+ Std. 923+ Teilnehmer, 74 Orte, 9 Länder	Video, Audio, 3D-Scans, Blickdaten, Stereo, Multi-Kamera, Text-Narrationen	Katalysator für die Forschung zur Ich-Perspektive; Benchmarks (Episodisches Gedächtnis, HOI, AV-Diarisierung, Soziales, Forecasting) [164]
EgoLife	300 Std. 6 Teilnehmer	egozentrisch, Interpersonell, Multiview, Multimodal, Audiotranskript, Visuell-auditive Narrationen	Grundlage für egozentrische Lebensassistenten, Frage-Antwort-Systeme mit langem Kontext [165]
Project Aria	Nicht spezifiziert (Forschungs-plattform)	Multimodale egozentrische Datenerfassung (Kameras, IMUs, GPS), SLAM, Blickerfassung, Hand-Tracking	egozentrische KI-Forschung, Computer Vision, Robotik, kontextbezogene KI [161]

Table 2.1: *Schlüsseldatensätze für egozentrische Vision und ihre Modalitäten*

Projekt zielt ebenfalls auf die Entwicklung eines ëgozentrischen Lebensassistentenäb, der Benutzer bei alltäglichen Aufgaben unterstützt, wie z.B. bei der Suche nach verlegten Gegenständen oder der Anleitung bei Reparaturen [165, 166].

Die praktischen Anwendungen sind vielfältig und umfassen die Gedächtnisunterstützung, bei der egozentrische Videoaufzeichnungen als Gedächtnisstütze dienen [164]. Ein weiterer Anwendungsfall ist die Aufgabenanleitung und das Erlernen von Fähigkeiten, bei denen Assistenten schrittweise Anleitungen für komplexe Vorgänge geben [166, 171]. Im Bereich der assistiven Technologien können Smart Glasses Texte vorlesen, Gesichter erkennen und vor Hindernissen warnen, was die Unabhängigkeit von Menschen mit Seh- oder Hörbehinderungen erheblich verbessert [170].

Zukünftige Systeme gehen über reaktive Hilfe hinaus und streben eine proaktive Unterstützung an. Blickbasierte KI kann die Absichten des Benutzers antizipieren, bevor explizite Befehle gegeben werden [173]. Das System Mirai liefert proaktive und kontextbezogene Anstöße zur positiven Verhaltensänderung durch kontinuierliche Umgebungsüberwachung [174]. Diese Entwicklung hin zu kontextbewussten, egozentrischen Systemen stellt einen Paradigmenwechsel in der Mensch-Maschine-Interaktion dar und verwirklicht die Vision von ubiquitären Assistenzdiensten.

System	Hauptfokus	Kernfunktionen	Zugrundeliegende Modelle/Datensätze
Vinci	Echtzeit-Assistent für verkörperte Interaktion	Szenenverständnis, temporale Verortung, Videozusammenfassung, Zukunftsplanung, kontextueller Chat, visuelle Aktionsgenerierung	EgoVideo-VL (Egocentric Vision Foundation Model + LLM) [171]
EgoLife	Egozentrischer Lebensassistent	Aktionserkennung, Narration, Frage-Antwort-Systeme, Verständnis langer Videos	EgoLife Dataset, EgoButler (EgoGPT + EgoRAG) [165]
Mirai	Proaktive kontextuelle Impulse zur Verhaltensänderung	Umgebungsanalyse, Absichtserkennung, personalisierte Interventionen	Tragbare Kamera, Echtzeit-Sprachverarbeitung [174]

Table 2.2: *Übersicht über egozentrische KI-Assistenzsysteme und deren Kernfunktionalitäten*

2.6.4 Zusammenfassungstechniken für egozentrische Video- und Audiodaten

Die Zusammenfassung von Videos ist entscheidend für die effiziente Verarbeitung langer egozentrischer Aufnahmen. Diese Videos stellen aufgrund ihrer kontinuierlichen Natur, starker Kamerabewegungen und häufiger Verdeckungen besondere Herausforderungen dar, die herkömmliche Techniken ungeeignet machen [160, 175, 176].

Es gibt extraktive, abstraktive und multimodale Ansätze. Extraktive Methoden wählen Schlüsselframes aus, während abstraktive Methoden neue Inhalte generieren [171]. Multimodale Ansätze integrieren Informationen aus verschiedenen Quellen für kontextreichere Zusammenfassungen [171].

Aktuelle Methoden umfassen ßtory-driven summarization", bei der kohärente Ketten von Subshots ausgewählt werden [175]. Trotz dieser Fortschritte zeigen spezialisierte Modelle bei egozentrischen Videos oft eine geringere Leistung. Interessanterweise übertrifft ein feinabgestimmtes Allzweckmodell wie GPT-4o einige dieser spezialisierten Modelle, was auf grundlegende Schwächen aktueller Ansätze hindeutet [171, 177].

2.6.5 Gesichtserkennungsalgorithmen und Anwendungen

Die Gesichtserkennung in egozentrischen Videos ist eine Kernfunktion für das Verständnis sozialer Interaktionen. Smart Glasses nutzen diese Technologie, um Personen zu identifizieren und den Benutzern, insbesondere solchen mit Sehbehinderungen, mehr Sicherheit in sozialen Situationen zu geben [170].

Die Herausforderungen sind erheblich. Verdeckungen stellen eines der hartnäckigsten Probleme dar und können die Erkennungsgenauigkeit drastisch reduzieren [160, 180]. Dynamische Blickwinkel und Kamerabewegungen erschweren die konsistente Erkennung zusätzlich [160].

Zur Bewältigung dieser Probleme werden fortschrittliche Deep-Learning-Modelle wie CNNs und GANs eingesetzt [180]. Edge Computing gewinnt an Bedeutung, um Latenzzeiten zu re-

Ansatz	Modelle	Stärken/Merkmale	Spezifische Herausforderungen
Extraktiv	Story-driven summarization [175], Unsupervised RL [176], TAC-SUM [171]	Wählt Schlüsselframes/-segmente aus, erfasst narrativen Fluss, anpassbar an Länge/Salienz	Hohe Redundanz, Kamerawackeln, schnelle Hintergrundwechsel, Verdeckungen [160]
Abstraktiv	Deep Neural Networks, NLP, Reinforcement Learning, GANs [178]	Generiert neue Inhalte, erfasst die Essenz, Potenzial für reichhaltigere Zusammenfassungen	Benötigt komplexe Modelle, hohe Rechenkosten [166]
Multimodal	EgoVideo-VL (Vinci) [171], GPT-4o (feinabgestimmt) [177]	Integriert Video, Audio, Text; umfassendes Verständnis; Fähigkeit von Allzweck-LLMs	Vollständige Nutzung reichhaltiger semantischer Inhalte, tiefe Fusion über Modalitäten hinweg [171]
Übergreifende Herausforderungen: Verständnis langer Videos, Personalisierung, Robustheit gegenüber egozentrischem Rauschen, Skalierbarkeit, Schwierigkeiten bei der Datenerfassung [166, 168, 179]			

Table 2.3: *Herausforderungen und Fortschritte bei der egozentrischen Videozusammenfassung*

duzieren und den Datenschutz zu erhöhen [181]. Angesichts der zunehmenden Verbreitung wachsen die Bedenken hinsichtlich des Datenschutzes. Als Reaktion darauf werden datenschutzfreundliche Techniken wie EgoBlur, MinusFace oder RLLFPR entwickelt [161, 182**?**].

2.6.6 Integration von Sprachbefehlen zur Steuerung von Assistenzdiensten

Sprachbefehle sind für die freihändige und intuitive Steuerung von KI-Assistenten auf egozentrischen Geräten von zentraler Bedeutung [170]. Die zugrunde liegende Technologie basiert auf KI und Natural Language Processing, um gesprochene Befehle zu interpretieren und auszuführen [183].

Ein wesentlicher Bestandteil ist die "Wake WordErkennung, die eine robuste Modellierung erfordert, um in realen Umgebungen mit Hintergrundgeräuschen zuverlässig zu funktionieren [184, 185]. Die Komplexität von Gesprächen mit mehreren Sprechern stellt nach wie vor eine große Herausforderung dar [186].

Trotz Fortschritten bleiben Probleme bestehen. Die Genauigkeit der Spracherkennung leidet in lauten Umgebungen, und die Interpretation verschiedener Akzente ist schwierig [186]. Auch die Verarbeitung von Umgangssprache stellt für NLU-Systeme eine Hürde dar [170]. Schließlich wirft der Älways-onModus erhebliche Datenschutzbedenken auf, was die Forschung an datenschutzfreundlichen Erkennungssystemen vorantreibt [185, 187].

Bereiche	Algorithmische Lösungen (State-of-the-Art)	Datenschutz / Techniken	Ziel / Auswirkung
Verdeckungen (selbst, extern, Accessoires) [160]	Deep-Learning-Modelle (CNNs, GANs, DenseNet, Inception) [180]	EgoBlur [161]	Verbesserte Genauigkeit trotz Hindernissen, Echtzeitleistung
Dynamische Blickwinkel/ Kamerawackeln [160]	Adaptive Lern- & Feedback-Mechanismen [181]	MinusFace [182]	Robustheit gegenüber Bewegung, verbesserte Reaktionszeiten
Variierende Lichtverhältnisse [180]	Edge Computing [181]	RLLFPR (Randomization and Local Feature Learning) [**?**]	Reduzierte Latenz, Datenschutz, Widerrufbarkeit
Begrenzte Daten für Verdeckungen [180]	Training auf diversen Datensätzen (Millionen von Bildern) [181]	—	Überwindung von Datenknappheit, hohe Genauigkeit

Table 2.4: *Schlüsselherausforderungen und neue Lösungen in der egozentrischen Gesichtserkennung*

2.6.7 Identifizierte Forschungslücken und zukünftige Richtungen

Eine zentrale Forschungslücke besteht in der Verfügbarkeit ausreichend großer, vielfältiger und kontinuierlicher egozentrischer Langzeitdatensätze. Bestehende Datensätze wie Ego4D gelten als unzureichend [160]. Die Erfassung solcher Daten ist arbeitsintensiv und wirft Datenschutzbedenken auf [168]. Zukünftige Forschung muss sich auf neue Benchmarks wie X-LeBench konzentrieren [188].

Die Robustheit der Modelle gegenüber dynamischen Blickwinkeln, Verdeckungen und der Notwendigkeit der Personalisierung ist ein weiterer kritischer Bereich. Aktuelle Ansätze haben oft Schwierigkeiten mit den spezifischen Herausforderungen egozentrischer Videos [160, 189]. Zukünftige Modelle müssen eine inhärente Robustheit aufweisen und implizite Signale wie den Blick des Nutzers integrieren [160, 189].

Die Echtzeitleistung und die Verarbeitung auf dem Gerät sind für die praktische Anwendung entscheidend. Die hohen Rechenanforderungen aktueller Modelle stellen eine erhebliche Einschränkung dar [166, 171]. Die Entwicklung effizienterer Modellarchitekturen und spezialisierter Hardware ist daher von großer Bedeutung [173].

Obwohl egozentrische Daten multimodal sind, wandeln viele Systeme diese Eingaben primär in Text um, was das Potenzial der reichhaltigen semantischen Informationen begrenzt [171]. Zukünftige Forschung sollte auf fortschrittliche multimodale Fusionsmodelle abzielen.

Schließlich werfen Älways-onGeräte tiefgreifende ethische Fragen auf. Massenüberwachung und mangelnde Transparenz sind ernsthafte Risiken [190]. Zukünftige Entwicklungen müssen "Privacy-by-DesignPrinzipien verfolgen und robuste, datenschutzfreundliche Techniken weiterentwickeln [161, 182, 190].

Kapitel 3

Implementierung und Methodik

3.1 Systemarchitektur

Das Ziel ist die Implementierung eines Systems, welches mittels Echtzeitdaten nicht nur latenzarme Sprache-zu-Sprache-Interaktion ermöglicht, sondern auch performanceoptimierte Bildanalyse, bei der Gesichter entweder erkannt oder sogar identifiziert und ausgeblendet werden. Um dies zu ermöglichen, werden verschiedene Komponenten benötigt, die alle ihren notwendigen Beitrag zum Gesamtsystem leisten.

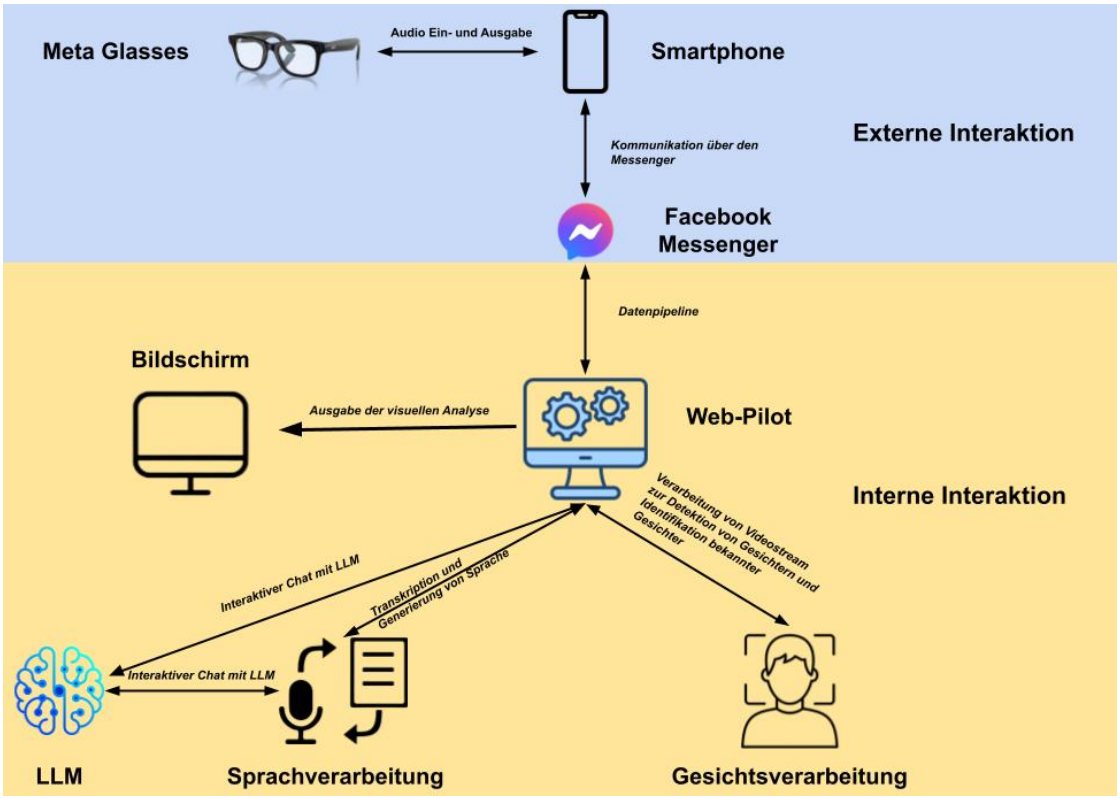


Figure 3.1: *Aufbau des Systems mit Unterteilung in externe und interne Interaktion*

Das System wird auf höchster Ebene in zwei Teile unterteilt, wie in 3.1 zu beobachten ist. Der obere Teil beschreibt die Interaktionen auf externer Ebene. Dabei sind alle gebräuchlichen Geräte und Programme gemeint, deren Nutzung Sinn und Zweck ermöglicht. Die für dieses Beispiel genutzten Meta Glasses dienen hierbei primär als Audio-Ein- und Ausgabe. Mit den in Kapitel 2 erwähnten technischen Features bietet die Brille die Möglichkeit, über das Mikrofon gesprochene Sprache aufzunehmen und gegebenenfalls über die integrierten Lautsprecher Ton abzuspielen. Darüber hinaus dient die integrierte Kamera als maßgeblicher Ursprung des Bildmaterials im Falle einer Bildübertragung.
Die Verbindung zum Netzwerk findet jedoch nicht smartphoneunabhängig statt. Letzteres bedient sich über Bluetooth an den eingehenden Daten und ist mit seiner Steuerung der Benutzeroberfläche als Steuergerät unerlässlich. Zudem wird für den Datenfluss die Facebook Messenger-App benötigt, welche sich nicht nativ auf der Brille befindet, sondern nur indirekt verbinden lässt.
Gegenüber den externen Interaktionen befindet sich nun der Kern des Systems, und zwar die interne Interaktion. In dieser finden alle Prozesse und Logikschritte statt, die Ergebnisse und Rechenschritte der Interaktion ermöglichen. Das zugrunde liegende System bedient sich von drei vorliegenden Komponenten. Angefangen mit eingehenden Nachrichten auf Textbasis im Chat, befindet sich am Ende der Kette das Large Language Model in Form einer ChatBot Python-Klasse. Eingehende Nachrichten des Nutzers werden entgegengenommen und mit dem zugrunde liegenden Modell sowohl verarbeitet als auch beantwortet. Die Antwort erfolgt über den direkten Weg zurück in den Chat.
Im Fall eines telefonischen Austauschs werden bestimmte Zwischenschritte benötigt. Dazu zählt, neben der Transkription der Sprache für ein besseres Verständnis, auch die Sprachgenerierung einer erhaltenen Antwort. Bei der letzten bedeutenden Komponente handelt es sich um die Gesichtsverarbeitung. In einer eigenen Klasse wird nach Auslösen des Verarbeitungssignals eingehendes Bildmaterial aufgearbeitet und analysiert. Ergebnisse werden neben der kommunikativen Ausgabe bei Bedarf auf dem Bildschirm in Form von Resultaten wiedergegeben.
Die auf diesem Level erklärten Elemente ermöglichen einen einwandfreien Datenfluss, der neben der Interaktion auch die Verarbeitung von auditiven und visuellen Daten ermöglicht.

3.2 Entwicklungsumgebung und eingesetzte Technologien

Neben der Logik des Systems sind die umstehenden Faktoren von großer Bedeutung. Dazu zählen neben der Hardware auch die verwendeten Softwarekomponenten. Wie in Kapitel zwei bereits erwähnt, eignet sich Python ideal. Durch seine Vielseitigkeit und Einfachheit in der Implementierung werden viele Bereiche ohne weitere Schwierigkeiten ermöglicht. Aufgrund einer großen Community und Dokumentation können Lösungen zu Problemen schnell gefunden werden. Außerdem sind Themengebiete wie Machine Learning und Big Data mit Abstand am meisten in speziellen Python-Bibliotheken vertreten. Dies macht es umso leichter, native Anwendungen und erste Ideen zu programmieren. Die Inklusion von objektorientierter Programmierung wird, wie später zu sehen sein wird, ein fundamentaler Bestandteil sein, der den Ablauf und die Exekution des Codes erst ermöglicht.
Viele der verwendeten Bibliotheken sind nur in Python erhältlich. Die hier relevanten Frameworks sind unter anderem Selenium für die Browserautomatisierung, Insightsface und OpenCV

zur Bildverarbeitung sowie Hugging Face zur Bereitstellung einiger Audio-Modelle. Die gesamte Entwicklung findet im kostenlosen OpenSource Code-Editor Visual Studio Code statt. Mit der integrierten Entwicklungsumgebung können leistungsstarke Module implementiert werden. Ein wesentlicher Bestandteil ist die Integration von Git. Für die Kontrolle von Versionsverläufen in komplexen Projekten kann die praktische Anwendung von Git auf GitHub gehostet werden und ist somit überall zugänglich.
Wie bereits zuvor erwähnt, gibt es neben den internen Softwarekomponenten externe Elemente, die maßgeblich für die Interaktion sind. Dazu zählt unter anderem die Brille von Meta. Mit der integrierten Kamera, dem Mikrofon und den Lautsprechern eignet sich das Gerät ideal, um eine reibungslose Integration in den Alltag zu ermöglichen. Die Wahl hat sich gerade aufgrund der qualitativ hochwertigen Kamera und des idealen Mikrofons durchgesetzt. Dadurch wird der Schritt in Richtung hochwertiger Datenverarbeitung erleichtert. Darüber hinaus lässt sich die Brille, wie zu Beginn erwähnt, durch Sprachbefehle steuern, wodurch unter anderem auch Anrufe an Personen ermöglicht werden. Dies eignet sich besonders gut, da die native Integration von Facebook Messenger in den vom gleichen Unternehmen hergestellten Nachrichtenservice bereits vorhanden ist. Die Wahl eines anderen Endgeräts wäre möglich, wird im Folgenden jedoch nicht weiter berücksichtigt. Die damit einhergehende Schlussfolgerung bezieht sich auf die Nutzung eines Smartphones, auf dem die Messenger App installiert ist. Idealerweise handelt es sich bei dem Smartphone um einen anderen Account als den, der später für den Web-Piloten genutzt wird.
Bei den letzten wichtigen Hardwarekomponenten handelt es sich um den Computer. Mit dem Ziel einer zukünftigen Skalierung auf Endgeräte war es das Ziel, so wenig Cloudrechenleistung wie möglich zu nutzen. Dies war bei der Nutzung von LLMs der Fall, wurde jedoch auf ein Minimum beschränkt. Sowohl die Benchmarks unterschiedlichster Modelle als auch die finale Ausführung fanden auf einem lokal installierten Rechner statt. Dieser beinhaltet eine CPU von AMD (AMD Ryzen 7 9700X mit 8 Kernen), 32 GB Arbeitsspeicher und eine Nvidia-Grafikkarte der Reihe 5700 Ti mit 16 GB VRAM. Die Wahl einer geeigneten Grafikkarte ist von großer Wichtigkeit, da die Ausführung vieler Modelle nicht auf jeder Grafikkarte ohne Probleme verläuft.
Nvidia ist im Bereich Machine Learning und insbesondere Deep Learning führend, vor allem aufgrund der speziellen Architektur ihrer Grafikprozessoren. Im Gegensatz zu CPUs sind diese für die parallele Verarbeitung von tausenden von Aufgaben ausgelegt. Mit einer enormen Menge an Daten ist dies ein besonderer Vorteil. Sowohl die Trainingszeiten neuer als auch die Inferenzzeiten bestehender Modelle lassen sich somit auf einen Bruchteil kürzen. Gerade Rechenzentren profitieren von der Multifunktionalität und einer energieeffizienten Arbeitsweise. Doch das Ökosystem profitiert gerade erst durch den Einsatz von zwei speziellen Technologien. Die Compute Unified Device Architecture (CUDA) ist eine Computing-Plattform. Mit dieser Reihe an Tools und APIs können letztere Grafikkarten für allgemeine Berechnungen genutzt werden. Viele beliebte Machine-Learning-Frameworks wie PyTorch und TensorFlow verwenden CUDA im Hintergrund, um mit der GPU zu kommunizieren und Berechnungen zu beschleunigen. Eine der wertvollsten Bibliotheken ist die CUDA Deep Neural Network Library (cuDNN). cuDNN bietet hochoptimierte Implementierungen für Standardoperationen, die häufig im Deep Learning vorkommen, wie z.B. Faltungsoperationen (Convolution), Pooling, Normalisierung und Matrixmultiplikation. Es ist entscheidend für die Beschleunigung von Deep-Learning-Frameworks und hilft, die Trainingszeit von Modellen erheblich zu ver-

kürzen. Zusätzlich verfügen NVIDIA-Grafikprozessoren über sogenannte Tensor Cores. Diese dedizierten Hardwarekomponenten sind spezifisch darauf ausgerichtet, die Rechenleistung für KI-Anwendungen und Deep-Learning-Prozesse massiv zu steigern. Sie sind für Matrixoperationen optimiert, die für das Training und die Inferenz von neuronalen Netzen grundlegend sind. Tensor Cores ermöglichen Mixed-Precision-Computing (z.B. FP16, TF32, FP8), was bedeutet, dass sie Berechnungen mit geringerer Präzision durchführen können, um die Geschwindigkeit zu erhöhen, während die Genauigkeit des Modells weitgehend erhalten bleibt. Dies führt zu erheblichen Leistungsverbesserungen gegenüber herkömmlichen CUDA-Kernen. Zusammenfassend lässt sich sagen, dass Nvidias Dominanz im Bereich des maschinellen Lernens auf der Kombination aus leistungsstarker, spezialisierter GPU-Hardware (mit CUDA Cores und Tensor Cores) und einem ausgereiften, breiten Software-Stack beruht, der Entwicklern die notwendigen Tools und Bibliotheken zur Verfügung stellt, um KI-Anwendungen effizient zu erstellen, zu trainieren und bereitzustellen.
Ein weiterer Faktor ist die Ein- und Ausgabe von Ton. Die Browserumgebung wird zwar in der Lage sein, Ton über das System und das Mikrofon aufzunehmen, benötigt jedoch Unterstützung bei der Eingabe der später erzeugten Audiodateien, sodass am anderen Ende die erzeugte Sprache hörbar wird. Dies wird mit sogenannten Virtual Audio Cables ermöglicht. Die Funktionsweise von Virtual Audio Cable basiert auf der Installation virtueller Audiogeräte, die vom Betriebssystem als vollwertige Soundkarten erkannt werden. Durch die interne Verbindung des Ausgangs eines virtuellen Kabels mit seinem Eingang können Audiodaten von einer Quellanwendung verlustfrei an eine Zielanwendung übertragen und dort weiterverarbeitet oder aufgezeichnet werden. Dadurch wird es möglich, das Text-to-Speech-Modul wirksam beim Endnutzer ankommen zu lassen.

3.3 Browser Automatisierung

Das Herzstück der Datenpipeline bildet der Browser. Dieser ermöglicht es, mit dem System zu interagieren. Dabei wird nicht nur Sprache, sondern auch Bild aufgenommen, verarbeitet und zurückgespielt. Um dies zu ermöglichen, müssen jedoch einige Bedingungen erfüllt sein. Dazu zählen neben dem Start des Browsers in einer virtuellen Umgebung auch die Navigation auf Facebook Messenger, das Akzeptieren der Cookies, das Anmelden mit den gegebenen Login-Daten, die Eingabe einer Messenger-spezifischen PIN zur Sicherheit und die aktive Dynamik zwischen Nachricht und Anruf. Dies alles findet unter der Bedingung statt, dass menschliches Verhalten imitiert wird, da es sich zwangsweise um gleiche Abläufe handeln muss, da keine offizielle Schnittstelle vorhanden ist.



Figure 3.2: *Methoden der Browser Klasse zum initialen Start*

Die in 3.2 zu sehenden Methoden sind notwendig, um alle zuvor erwähnten Punkte abzuhaken. Eine Besonderheit liegt hierbei in der Art und Weise, wie mit gewissen Sicherheitsmaßnahmen der Website umgegangen wird. Aufkommende Elemente werden mittels Klicks oder Tastensimulationen weggeklickt, wodurch dem Browser gegenüber der Eindruck menschlicher

Interaktion vermittelt wird. Zuerst wird der Button mit dem Text Älle Cookies akzeptieren"gesucht. Dabei wird der Suche ein gewisses Maß an Geduld zugewiesen, indem abgewartet wird, bis der Button erscheint. Dies ist jedoch nur nach aufmerksamer Untersuchung möglich, da es sonst zu langem Warten führen würde. Sobald das Element gefunden wurde, wird ein Klick darauf ausgeführt und anschließend eine Bestätigung angezeigt. Das Finden des "Cookies akzeptieren"Buttons sieht dementsprechend folgendermaßen aus:

```
def accept_cookies(self):
    try:
        allow_cookies_button_xpath = "//button[contains(., 'Alle Cookies erlauben')]"
        cookie_button = self.wait.until(EC.element_to_be_clickable((By.XPATH, allow_cookies_button_xpath)))
        time.sleep(3)
        cookie_button.click()
        print("Clicked on 'Alle Cookies erlauben' button.")
    except Exception as e:
        print(f"Could not find or click the 'Alle Cookies erlauben' button: {e}")
```

Listing 3.1: Ïnteraktion mit Website Elementen"

Neben der Suche und dem Klick auf Elemente ist auch die Eingabe von Texten zur Navigation notwendig. Gerade beim Login oder bei der Eingabe des Authentifizierungspins ist eine Lösung erforderlich. Mit einem send-keys-Befehl wird dafür Hilfe geschaffen.

```
email_field = self.driver.find_element(By.ID, "email")
password_field = self.driver.find_element(By.ID, "pass")

for char in self.email:
    email_field.send_keys(char)
    time.sleep(0.3)
for char in self.password:
    password_field.send_keys(char)
    time.sleep(0.3)
```

Listing 3.2: SSimulation der Tastatureingabe"

Nachdem die notwendigen Schritte erfüllt wurden, bleiben nun alle Methoden offen, die für die dynamische Interaktion verantwortlich sind.

Für einen kontextabhängigen Chat muss mit vorliegenden Nachrichten die Basis geschaffen werden. Ohne die Information, welche Sprache welche Nachricht zuletzt von sich gegeben hat, würde es nur zu Verwirrungen und Logikfehlern beim Large Language Model führen. Dafür ist die Chat-Funktion da. Sie untersucht den Chat auf die letzten Nachrichten und ordnet neue Nachrichten dem Kontext zu. Nach Erhalt, Verarbeitung und Antwort gelangt Letztere über text_antwort zurück in den Chat. Dabei wird der Text auf die gleiche Art und Weise wie zuvor über simulierte Tastenanschläge eingegeben, wobei jeder Buchstabe mit einer zeitlichen Differenz eingegeben wird.
Neben der aktiven Konversation im Chat ist es ebenfalls von Bedeutung, eingehende Anrufe sowohl anzunehmen als auch gegebenenfalls zu beenden. Mit den Funktionen annehmen und schließen wird in einer Schleife auf eingehende Änderungen in der Browserumgebung reagiert. Während eines Telefonats wird gewartet, bis ein Element "Qualität des Anrufs bewertenäuftaucht, was letztendlich das Zeichen für das Ende des Telefonats ist. Da sich mit

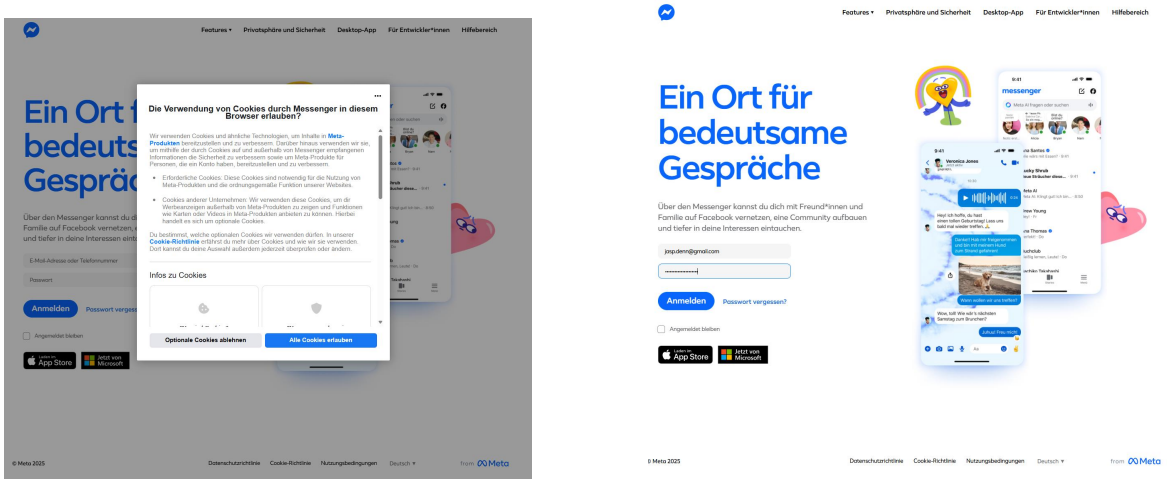


Figure 3.3: *Initialer Screen zur Eingabe nach Start des Browsers*



Figure 3.4: *Methoden der Browser Klasse für Interaktion und Verarbeitung*

einem eingehenden Anruf ein zusätzliches Fenster öffnet, muss dieses auch wieder geschlossen werden können. Das schafft close_browser, in dem der Wechsel auf das ursprüngliche Fenster stattfindet. Der Browser ist in der Lage, das Mikrofon und die Kamera des Systems zu nutzen.

Eine bisher nicht erwähnte Methode betrifft die Verarbeitung des Videostreams. In einem Videocall erscheint das aufgenommene Bild des Endnutzers Frame für Frame im künstlichen Browser. Um hieraus einen möglichst flüssigen Stream zu erreichen, werden mithilfe des Browsers Screenshots gemacht. Letztere entstehen jedoch nicht bildschirmübergreifend, sondern passen sich der Position des Browsers an, um möglichst präzise arbeiten zu können und performant zu wirken.

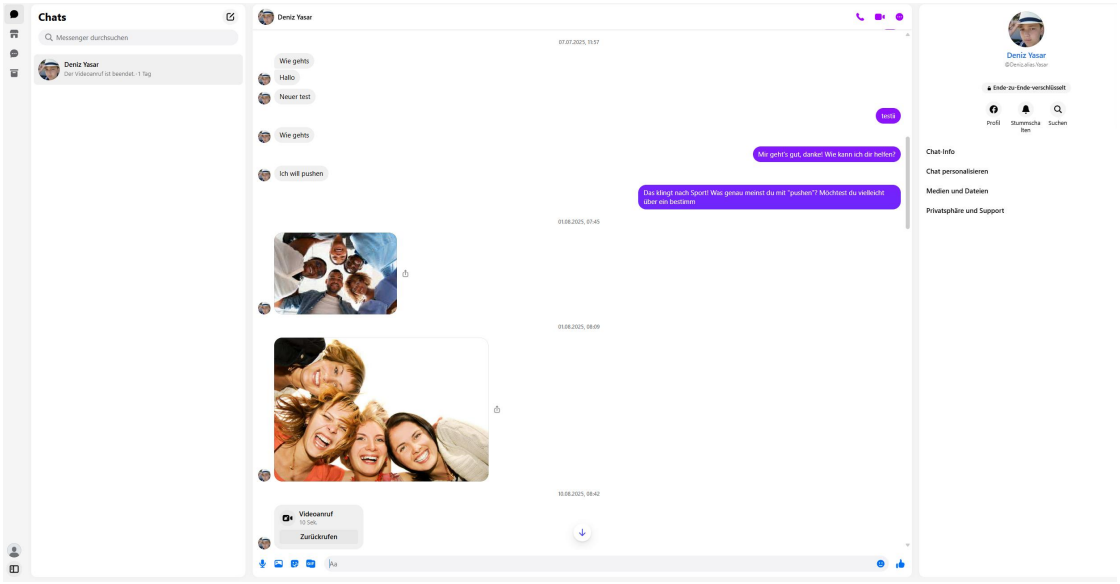


Figure 3.5: *State nach erfolgreichem Login*

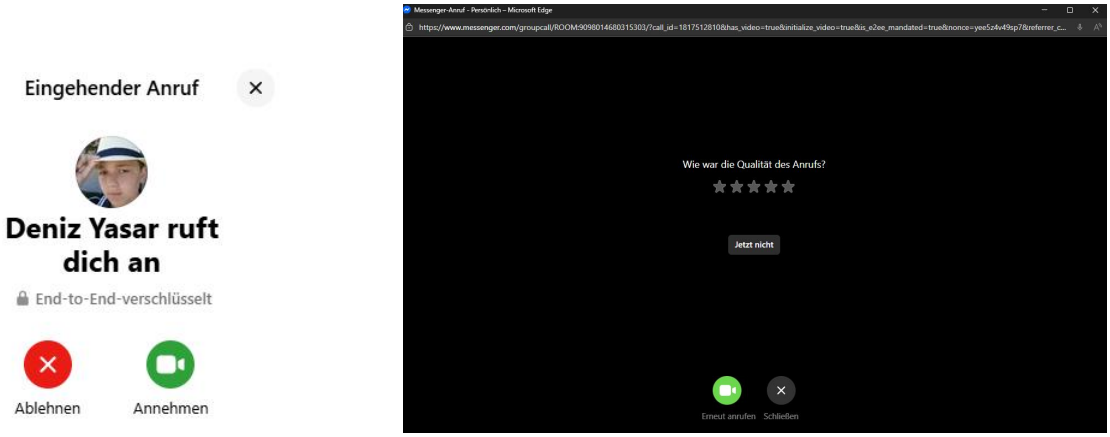


Figure 3.6: *Interaktive Pop-Up Elemente und Fenster bei eingehendem Anruf*

3.4 Chatbot

Das Large Language Model ist in der Chatbot-Klasse unumgänglich. Dieses ist nicht nur für eine reibungslose Konversation im Chat zuständig, sondern auch für Antworten in bestimmten Formaten, wie gleich gezeigt werden wird.
Neben einfachen Unterhaltungen wird das LLM trainiert, Ansprüche und Wünsche der Nutzer zu erkennen und wiederzugeben. Dies erreicht man durch Prompt-Engineering. Prompt-Anweisungen sind von entscheidender Bedeutung, da sie als präzise Anleitungen einem KI-System ermöglichen, unsere Absichten und den Kontext einer Aufgabe exakt zu verstehen. Sie dienen dazu, den potenziellen Lösungsraum klar abzugrenzen und die generativen Prozesse des Systems gezielt zu steuern, sodass die ausgegebene Information nicht zufällig, sondern hochrelevant und passgenau ist. Im Grunde genommen stellen sie die formale Spezifikation dar, mit der wir das Modell über das gewünschte Ergebnis informieren, einschließlich des Zwecks, des Formats, des Stils und jeglicher Einschränkungen. Die sorgfältige Formulierung dieser Anweisungen, oft als "Prompt Engineering"bezeichnet, ist daher der Schlüssel, um die Leistungsfähigkeit und Verlässlichkeit von KI-Systemen optimal zu nutzen. Kurz gesagt, durch präzise Anweisungen lehren wir die KI, genau das zu liefern, was wir brauchen, und schöpfen so ihr volles Potenzial aus.
Die Wahl einer cloud-basierten Lösung fällt auf die Anbindung des Google's Gemini-2.5-Flash-Lite Modells. Nähere Gründe für die Wahl werden in Kapitel 4 erläutert. Die API fällt eher simpel aus und ermöglicht einen schnellen Zugriff.

```
generation_config = {
    "temperature": 0.7,
    "max_output_tokens": 100,
    "top_p": 1.0,
    "top_k": 32,
}
self.model = genai.GenerativeModel('gemini-2.5-flash-lite',
    generation_config=generation_config)
self.history = []

self.chat = self.model.start_chat(history=[
    {'role': 'user', 'parts': [self.anweisung]},
    {'role': 'model', 'parts': ["Understood. I will follow these instructions
    ."]}
])
```

Listing 3.3: "Verbindung und Parametereinstellung von Gemini 2.5 Flash Lite"

Die effiziente Steuerung generativer Sprachmodelle erfordert ein Verständnis spezifischer Konfigurationsparameter, die das Verhalten und die Ausgabe dieser Systeme maßgeblich beeinflussen. Die hier besprochenen Einstellungen – temperature, max_output_tokens, top_p und top_k – sind grundlegend für die Gestaltung der generierten Inhalte hinsichtlich ihrer Zufälligkeit, Länge und Wortwahl. Der Parameter temperature reguliert die Kreativität oder Zufälligkeit der vom Modell erzeugten Ausgabe. Ein Wert von 0,7, wie hier angegeben, suggeriert eine mittlere Balance: Höhere Werte fördern die Auswahl weniger wahrscheinlicher, aber potenziell originellerer Wörter, während niedrigere Werte die Ausgabe vorhersehbarer und konservativer gestalten. Diese Einstellung ist entscheidend für die Variabilität und den Ideenreichtum des generierten Textes. max_output_tokens definiert die obere Grenze der

Ausgabelänge und ist ein direktes Maß für die Quantität des erzeugten Textes. Eine Einstellung von 100 bedeutet, dass das Modell seine Generierung stoppt, sobald es 100 Tokens – das sind grundlegende Textbausteine wie Wörter oder Satzzeichen – erreicht hat. Dieser Parameter ist essenziell für die Ressourcenkontrolle und die Begrenzung der Antwortgröße. Die Parameter top_p und top_k beeinflussen die Auswahl des nächsten Wortes im Generierungsprozess. top_p, auch als Nucleus Sampling bekannt, bestimmt, welche Wörter auf Basis ihrer kumulierten Wahrscheinlichkeit in die engere Wahl kommen. Ein Wert von 1.0, wie hier, erlaubt dem Modell, alle potenziellen nächsten Wörter zu berücksichtigen. Im Gegensatz dazu beschränkt top_k die Auswahl des nächsten Tokens auf die K wahrscheinlichsten Kandidaten. Ein top_k von 32 bedeutet demnach, dass das Modell nur aus den 32 wahrscheinlichsten Wörtern auswählt. Diese beiden Parameter wirken oft zusammen, um eine kontrollierte Balance zwischen Offenheit für neue Ideen und der Einhaltung von Kohärenz bei der Textgenerierung zu gewährleisten. Das zu Beginn der Sektion erwähnte Prompt-Engineering wird hier unter self.anweisung verwendet. Im Grunde benötigen wir aus der Antwort des Sprachmodells ein gewisses Format. Für das System ist es wichtig zu verstehen, in welcher Sprache gesprochen wird, was die Antwort ist und ob eine Funktion erkannt wurde. Unter den Funktionen finden sich die in der Gesichtsverarbeitung wichtigen Anweisungen detection, recognition und count wieder. Hier ist ein Ausschnitt des initialen Prompts zu sehen:

```
1 detected language in shortform like en for english or de for german or fr for
      french ; detected function ; bot response
2 The detected language short is the language the user is using in the
     conversation.
3 the detected function is the function that the user wants you to execute, if
     no functions was identified from the known then write n/a.
```

Listing 3.4: "Prompt Engineering der gewünschten Ausgaben"

Ein möglicher Output auf Ërkenne mir alle Gesichter"wäre dementsprechend 'de ; recognition ; Ja, natürlichöder auf "Who is the current president of the United States?"dementsprechend ën ; n/a ; The current President is Donald J. Trump". Mit den Methoden get_completion und chatting können dementsprechend erhaltene Nachrichten versandt und wiedergegeben werden. Um den Konversationsverlauf zu speichern und griffbereit zu gestalten, gibt es mit folgender Zeile die Garantie, dass Nachrichten in den Verlauf eingebettet werden.

```
1 self.history.append({"role": "user", "content": user_input})
2 self.history.append({"role": "assistant", "content": assistant_response})
```

Listing 3.5: SSicherstellung des kontextabhängigen Verlaufs"

Mit den implementierten Ansätzen ist nicht nur eine automatisierte Steuerung des Browsers möglich, sondern auch die Verarbeitung und Analyse von Text in einer Konversation. Bislang ist jedoch nur die einwandfreie Konversation im Chat möglich. Um den multimodalen Ansprüchen gerecht zu werden, benötigt es darüber hinaus eine Klasse, die sich mit der Sprachverarbeitung beschäftigt.

3.5 Sprachverarbeitung

Das Ziel der Klasse RealTimeVoiceProcessor ist die auditive Verarbeitung von eingehender Sprache und die Übersetzung des Ergebnisses des Large Language Models in abhörbare Form,

unter der Bedingung geringer Latenz. Um dies zu erreichen, werden neben der Anbindung des Chatbots auch ein Modell zur Transkription und Sprachgenerierung benötigt. Für die Speech-to-Text-Pipeline wird das fast-whisper Medium-Size Modell in Erwägung gezogen. Aufgrund seiner Qualität und Schnelligkeit eignet es sich ideal für den hier genutzten Zweck. Daneben ist für den Text-to-Speech Part FastPitch ideal geeignet. Mit seiner Stärke in Geschwindigkeit und authentischer Stimmlage wird sichergestellt, dass eine angenehme Konversation stattfinden kann.

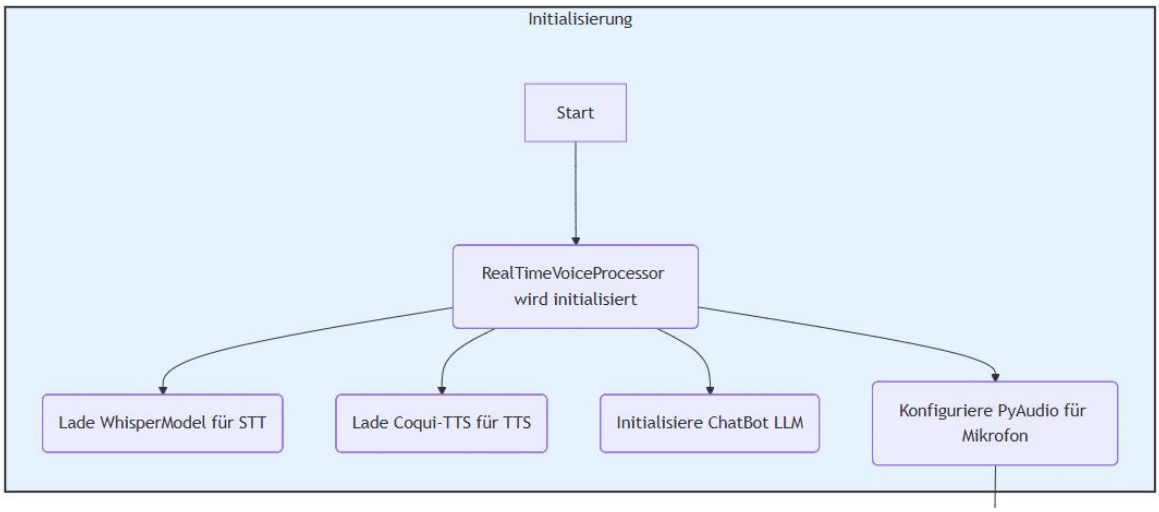


Figure 3.7: *Alle notwendigen Elemente zur Verarbeitung von Sprache*

Die Funktionsweise des Systems lässt sich in zwei logisch getrennte, aber sequenziell abhängige Phasen unterteilen: die Systeminitialisierung und Ressourcenallokation sowie die operative Hauptschleife, welche die kontinuierliche Verarbeitung der Audiosignale übernimmt. Diese Trennung ist fundamental, um eine minimale Latenz während der interaktiven Phase zu gewährleisten. Die Initialisierungsphase ist als vorbereitender, einmaliger Prozess konzipiert, der die Allokation aller notwendigen Hard- und Softwareressourcen vor dem Eintritt in den interaktiven Modus sicherstellt. Ein zentraler Aspekt dieser Phase ist das proaktive Laden der rechenintensiven Modelle für künstliche Intelligenz in den Arbeitsspeicher. Konkret umfasst dies die Instanziierung eines hochperformanten Speech-to-Text-Modells der Whisper-Klasse, das für die präzise Transkription akustischer Signale in Textform verantwortlich ist, sowie eines Text-to-Speech-Modells aus der Coqui-TTS-Bibliothek, welches die umgekehrte Transformation von Text in eine synthetische, natürlich klingende Sprachausgabe ermöglicht. Gleichzeitig wird die Verbindung zum Kern der logischen Verarbeitung, einem als ChatBot gekapselten LLM, etabliert. Ein weiterer kritischer Schritt ist die Konfiguration der Audio-Eingabe-Pipeline über die PyAudio-Bibliothek. Diese agiert als Abstraktionsschicht zur Hardware und ermöglicht eine plattformunabhängige Ansteuerung des Standard-Mikrofons, dessen Parameter wie Abtastrate und Kanalanzahl für die Spracherkennung optimiert werden. Von besonderer Bedeutung ist hierbei die dynamische Prüfung auf verfügbare Hardwarebeschleuniger, insbesondere CUDA-fähige Grafikeinheiten. Die Auslagerung der Tensor-Operationen der KI-Modelle auf eine GPU ist entscheidend, um die Verarbeitungszeiten von Sekunden auf Millisekunden zu reduzieren und somit die Voraussetzung für eine flüssige Konversation zu

schaffen. Nach Abschluss dieser Phase sind alle Komponenten vorkonfiguriert und betriebsbereit, wodurch Latenzen durch dynamisches Nachladen während des Betriebs vermieden werden. Nach Initialisierung tritt das System in seine operative Hauptschleife ein, die als Herzstück der Echtzeitverarbeitung fungiert. Diese Schleife implementiert einen kontinuierlich laufenden Prozess, der im Wesentlichen als eine Zustandsmaschine mit den Zuständen des Zuhörens, Aufnehmens und Verarbeitens beschrieben werden kann. Der Kernmechanismus dieser Schleife ist eine Form der Sprechpausenerkennung, die als effizienter Filter dient, um relevante Sprachsignale von Stille oder irrelevanten Umgebungsgeräuschen zu segmentieren. In sehr kurzen Intervallen wird ein kleiner Datenblock, ein sogenannter Chunk, aus dem Audiostream des Mikrofons gelesen und einer Amplitudenanalyse unterzogen. Solange die gemessene Lautstärke unter einem definierten Schwellenwert verbleibt, verharrt das System im passiven Zuhörmodus. Überschreitet die Amplitude diesen Wert, wird dies als Beginn einer sprachlichen Äußerung interpretiert, woraufhin das System in den Aufnahmezustand wechselt und die nachfolgenden Audio-Chunks in einem Puffer akkumuliert. Die Beendigung der Äußerung wird durch das Unterschreiten des Schwellenwerts für eine spezifische, vordefinierte Zeitdauer detektiert. Erst dieses Ereignis triggert die vollständige Verarbeitungspipeline: Die gepufferten Audiodaten werden zu einer temporären Audiodatei zusammengefügt und der automatischen Spracherkennung zugeführt. Der daraus resultierende Text wird zur semantischen Analyse und zur Generierung einer kohärenten Antwort an das LLM weitergeleitet. Die textuelle Antwort des Modells wird abschließend mittels Sprachsynthese in eine akustische Wellenform umgewandelt und über die Audioausgabe wiedergegeben. Dieser ereignisgesteuerte Ansatz stellt sicher, dass die rechenintensiven Operationen nur bei einer abgeschlossenen und relevanten Spracheingabe ausgeführt werden, was die Systemressourcen schont und eine hohe Reaktionsfähigkeit gewährleistet.
Abbildung 3.8 verdeutlicht den Ablauf und zeigt auf, in welchem Grundprinzip die Klasse arbeitet, sobald die Aufnahme des Tons beendet ist. Dies ermöglicht einen reibungslosen Ablauf hinsichtlich der Audioaufnahme, -verarbeitung und -ausgabe.

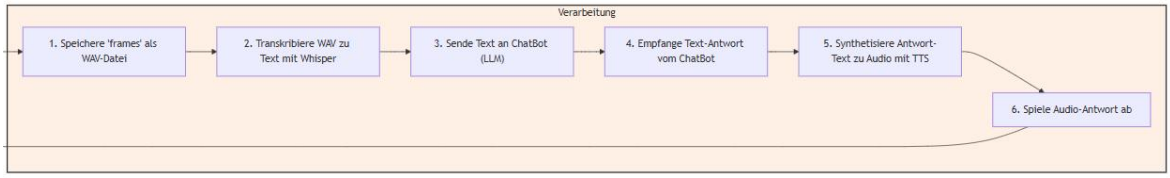


Figure 3.8: *Verarbeitungsschleife der Sprache*

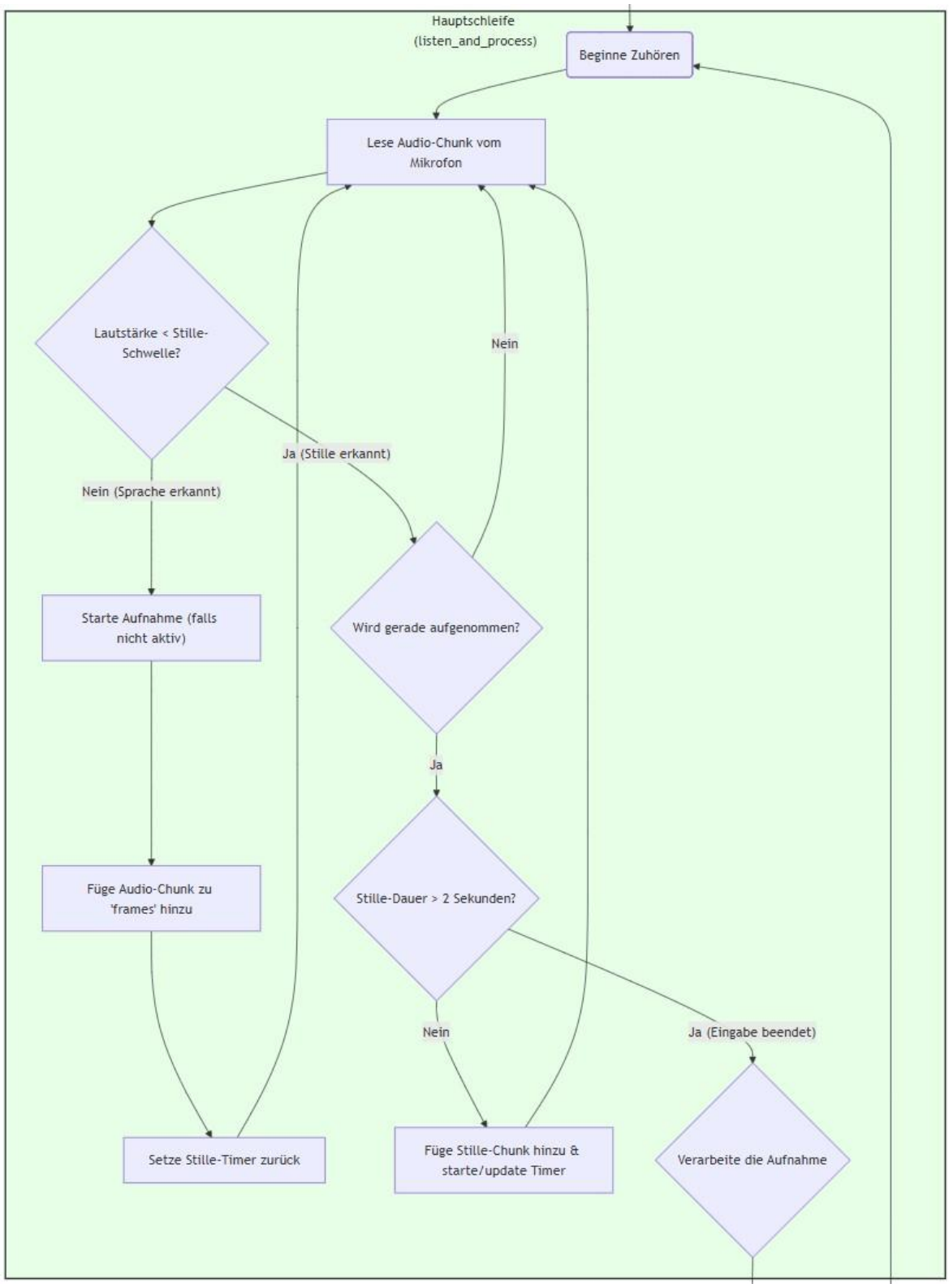


Figure 3.9: *Hauptschleife der Audioaufnahme mit Fokus auf aktive Stimmerkennung durch Lautstärkeschwellen*

3.6 Gesichtsverarbeitung

Für die vollständige Analyse des Bildmaterials und der damit verknüpften Funktionen ist ein Blick in die Architektur notwendig. Das Ziel ist es, die Verarbeitung von Gesichtern so recheneffizient wie möglich zu integrieren. Zur Veranschaulichung der internen Prozessarchitektur der Klasse wird ein Flussdiagramm herangezogen. Dieses Diagramm illustriert den sequenziellen Datenfluss und die logischen Verzweigungen innerhalb des Systems, beginnend bei der Rohdatenerfassung bis hin zur finalen Visualisierung. Es dient dem besseren Verständnis, wie die Klasse ihre hohe Effizienz und Echtzeitfähigkeit durch eine durchdachte Kombination aus nebenläufiger Verarbeitung und optimierten Analysemethoden erreicht.

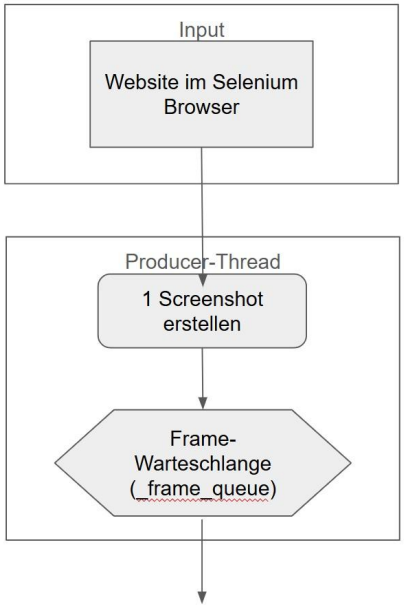


Figure 3.10: *Initialer Start der Bereitstellung des Browserfenster-Inhalts*

Der Prozess beginnt mit der Datenerfassung, die in einem dedizierten Thread, dem sogenannten Producer, ausgeführt wird. Dieser Thread greift mittels eines Selenium-WebDrivers auf eine aktive Webseite zu und erstellt in kurzen, regelmäßigen Abständen einen Screenshot. Diese Vorgehensweise stellt ein besonderes Merkmal der Architektur dar: die strikte Trennung von I/O-lastigen Operationen (Bilderfassung) und rechenintensiven Aufgaben (Analyse). Der erfasste Screenshot wird als Rohdatensatz in eine Puffer-Warteschlange (_frame_queue) gelegt. Diese Warteschlange fungiert als asynchrones Bindeglied zwischen dem Erfassungs- und dem Analyseprozess und verhindert, dass einer der beiden Threads auf den anderen warten muss, was Latenzen minimiert. Ein zweiter, parallel laufender Thread, der Consumer, entnimmt kontinuierlich die Bilddaten aus dieser Warteschlange für die weitere Verarbeitung. Nach der Entnahme wird das Bild zunächst dekodiert und auf eine vordefinierte Analysebreite skaliert. Diese Normalisierung der Bildgröße gewährleistet eine konsistente und performante Verarbeitung, unabhängig von der ursprünglichen Auflösung des Screenshots. Anschließend erreicht der Datenfluss den zentralen Entscheidungsknoten der Pipeline, der den weiteren Verarbeitungsweg anhand des aktuell konfigurierten Betriebsmodus (mode) bestimmt. Ist der Modus auf detection gesetzt, verfolgt das System einen simplifizierten Pfad. Die InsightFace-

Bibliothek wird hierbei ausschließlich zur Lokalisierung von Gesichtern im Bild genutzt. Jedes detektierte Gesicht wird im Anschluss mit einem visuellen Indikator, typischerweise einem blauen Rechteck, auf dem Ausgabebild markiert. Dieser Modus ist auf maximale Geschwindigkeit bei der reinen Anwesenheitserkennung ausgelegt. Wird hingegen der anspruchsvollere recognition-Modus verwendet, durchläuft das Bild einen erweiterten Analyseprozess. In einem ersten Schritt führt InsightFace eine kombinierte Operation aus Detektion und Merkmalsextraktion durch.

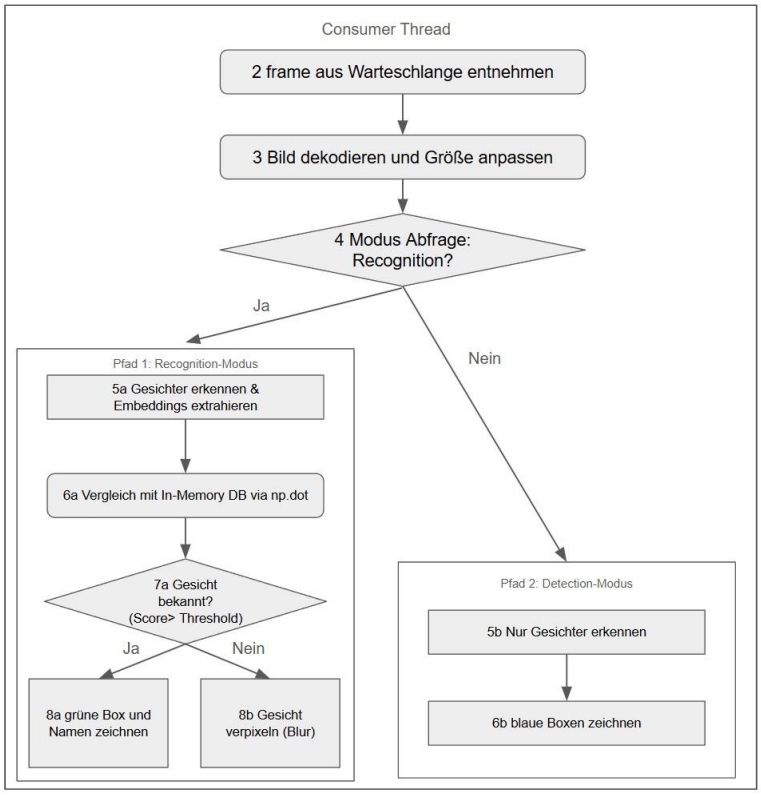


Figure 3.11: *Prozesswahl der Klasse in Abhängigkeit des gewählten Modus*

Für jedes erkannte Gesicht wird ein hochdimensionaler Merkmalsvektor, ein sogenanntes Embedding, generiert, der die einzigartigen Charakteristika des Gesichts numerisch repräsentiert. Dieses extrahierte Embedding wird normalisiert und anschließend mit einer im Arbeitsspeicher vorab geladenen Datenbank bekannter Gesichts-Embeddings verglichen. Ein wesentliches Leistungsmerkmal der Klasse ist die Implementierung dieses Vergleichs durch eine hochoptimierte, vektorisierte Skalarprodukt-Berechnung mittels NumPy. Dieser Ansatz ermöglicht den simultanen Abgleich mit potenziell Tausenden von Referenzbildern in minimaler Zeit. Basierend auf dem höchsten resultierenden Ähnlichkeitswert (Score) wird eine Entscheidung getroffen: Überschreitet der Wert einen vordefinierten Schwellenwert (recognition_threshold), gilt das Gesicht als identifiziert. Es wird mit einem grünen Rechteck sowie dem zugehörigen Namen versehen. Liegt der Wert darunter, wird das Gesicht als unbekannt klassifiziert und zum Schutz der Privatsphäre durch einen starken Weichzeichner anonymisiert. Beide Verarbeitungspfade konvergieren schließlich in der finalen Visualisierungsphase.

Dem bearbeiteten Bild werden zusätzliche Metadaten wie der aktuelle Modus und die Verarbeitungsgeschwindigkeit in Bildern pro Sekunde hinzugefügt. Das vollständig annotierte Bild wird daraufhin in einem Fenster zur Echtzeit-Anzeige ausgegeben. Das Flussdiagramm verdeutlicht somit eine Architektur, die durch Parallelisierung, spezialisierte Modi und algorithmische Optimierungen gezielt für anspruchsvolle Echtzeitanwendungen konzipiert wurde.

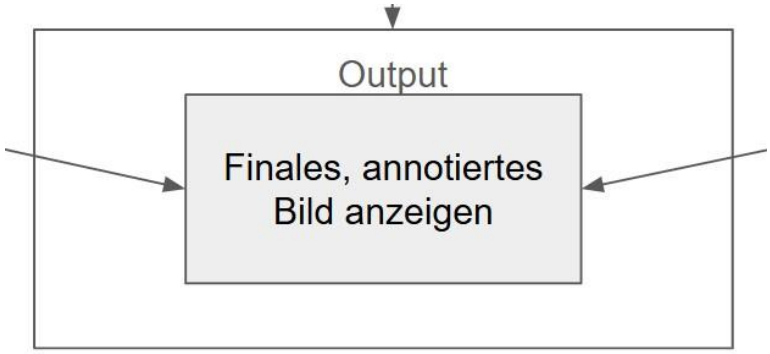


Figure 3.12: *Zusammenführung des Entscheidungbaums zur finalen Analyse per Bild und Ton*

Um die dynamischen Interaktionen und die zeitliche Abfolge der Kommunikation zwischen den nebenläufigen Komponenten der Klasse zu analysieren, wird ein Sequenzdiagramm herangezogen. Während das zuvor beschriebene Flussdiagramm den logischen Pfad der Datenverarbeitung darstellt, fokussiert sich das Sequenzdiagramm auf das zeitliche Zusammenspiel und den Nachrichtenaustausch zwischen den beteiligten Objekten und Threads. Es ist somit unerlässlich, um das Kernprinzip der Architektur – das Producer-Consumer-Muster – und dessen Beitrag zur Systemleistung und Stabilität zu verstehen. Die zentralen Akteure, die in diesem Diagramm als vertikale Lebenslinien dargestellt werden, sind der Haupt-Thread (MainThread), der die Steuerung innehat, der Erfassungs-Thread (CaptureThread), der Analyse-Thread (AnalysisThread) sowie die als gemeinsamer, synchronisierter Puffer dienende Warteschlange (FrameQueue). Die Interaktion wird durch den MainThread initiiert, indem er die öffentliche Methode start_analysis() aufruft. Dieser Aufruf bewirkt die Instanzierung und den Start der beiden Worker-Threads, CaptureThread und AnalysisThread, welche daraufhin ihre autonomen Verarbeitungsschleifen beginnen. Nach der Initialisierung treten die beiden Worker-Threads in ihre jeweiligen, parallel ablaufenden Operationszyklen ein, die das Herzstück der Echtzeitverarbeitung bilden. Der CaptureThread agiert als Producer. In seiner Schleife sendet er periodisch eine Anfrage an den Selenium-WebDriver zur Erstellung eines Screenshots. Die resultierenden Bilddaten werden anschließend durch einen put()-Aufruf als Nachricht an die FrameQueue übergeben. Dieser Vorgang ist nicht-blockierend, sodass der CaptureThread unmittelbar mit der nächsten Erfassung fortfahren kann, solange die Warteschlange nicht ihre maximale Kapazität erreicht hat. Parallel dazu agiert der AnalysisThread als Consumer. Seine Schleife beginnt mit einem get()-Aufruf an die FrameQueue. Hierbei handelt es sich um einen blockierenden Aufruf: Der Thread verharrt in einem ressourcenschonenden Wartezustand, bis ein neues Bildelement in der Warteschlange verfügbar ist. Sobald Daten empfangen werden, setzt der Thread seine Ausführung fort und führt die rechenintensive Analyse (Detektion oder Erkennung) durch. Diese Entkopplung ist ein entscheidendes

Leistungsmerkmal der Klasse. Sie stellt sicher, dass die Analysegeschwindigkeit die Bilderfassungsrate nicht direkt beeinflusst und umgekehrt, wodurch Engpässe und Latenzspitzen effektiv vermieden werden. Das Sequenzdiagramm visualisiert ebenso den Prozess der kontrollierten Terminierung. Durch den Aufruf von stop_analysis() im MainThread wird ein internes Event-Flag gesetzt. Dieses Ereignis dient als Signal für beide Worker-Threads, ihre Verarbeitungsschleifen zu beenden. Die Threads prüfen dieses Flag bei jeder Iteration und terminieren sich selbstständig. Schließlich zeigt das Diagramm, dass der MainThread mittels join()-Aufrufen auf die vollständige Beendigung beider Threads wartet. Dies garantiert einen sauberen und geordneten Systemabschluss ohne verwaiste Prozesse.

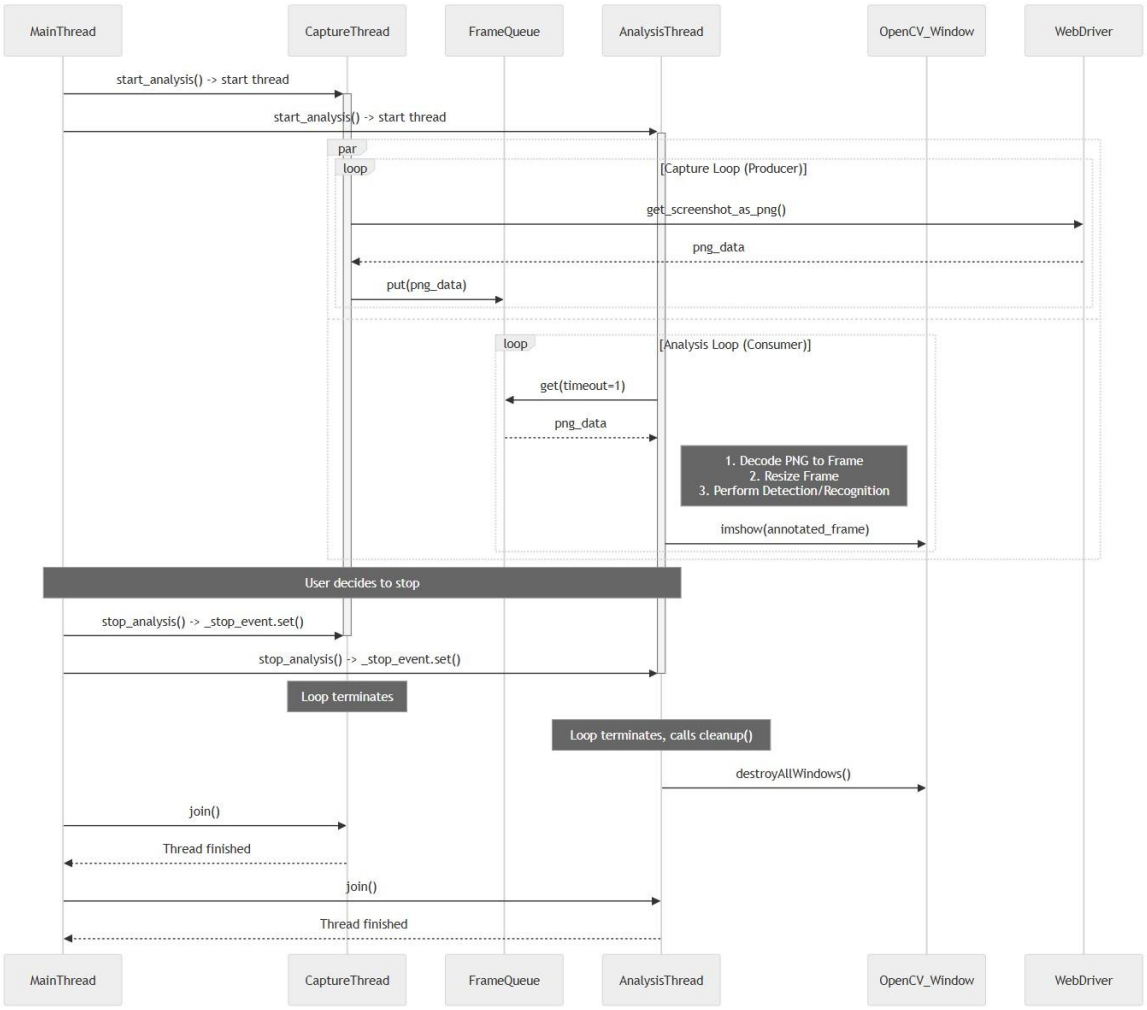


Figure 3.13: *Zusammenspiel der Komponenten zur Optimierung der Rechenleistung*

Zur Vervollständigung der Systemanalyse und zur Darstellung der internen Verhaltenslogik der FullyOptimizedRealTimeFaceProcessor-Klasse wird ein Zustandsdiagramm eingesetzt. Dieses Diagramm abstrahiert von den konkreten Datenflüssen und Thread-Interaktionen und konzentriert sich stattdessen auf den Lebenszyklus des Prozessor-Objekts, seine diskreten

Betriebszustände sowie die Ereignisse, die Übergänge zwischen diesen Zuständen auslösen. Es bietet somit eine essenzielle Makroperspektive auf die Steuerbarkeit und die definierten Verhaltensmodi des Systems. Der Lebenszyklus des Objekts beginnt unmittelbar nach seiner Instanziierung im initialen Zustand Gestoppt. Dieser Zustand repräsentiert einen inaktiven, aber betriebsbereiten Modus, in dem keine Threads aktiv sind und keine Ressourcen für die Bildverarbeitung beansprucht werden. Das System verbleibt in diesem Ruhezustand, bis ein externes Ereignis einen Zustandswechsel anstößt. Der primäre Übergang erfolgt vom Zustand Gestoppt in den aktiven Zustand Läuft. Dieser wird durch den Aufruf der öffentlichen Methode start_analysis() ausgelöst. Mit diesem Ereignis werden die beiden Worker-Threads (CaptureThread und AnalysisThread) gestartet, und die kontinuierliche Verarbeitung von Bilddaten beginnt. Der Zustand Läuft ist jedoch nicht monolithisch, sondern als Kompositzustand konzipiert, der die beiden zentralen Betriebsmodi der Klasse als innere Zustände kapselt: den Detection-Modus und den Recognition-Modus. Beim Eintritt in den Läuft-Zustand wird standardmäßig der zuletzt konfigurierte oder der initiale Modus eingenommen.

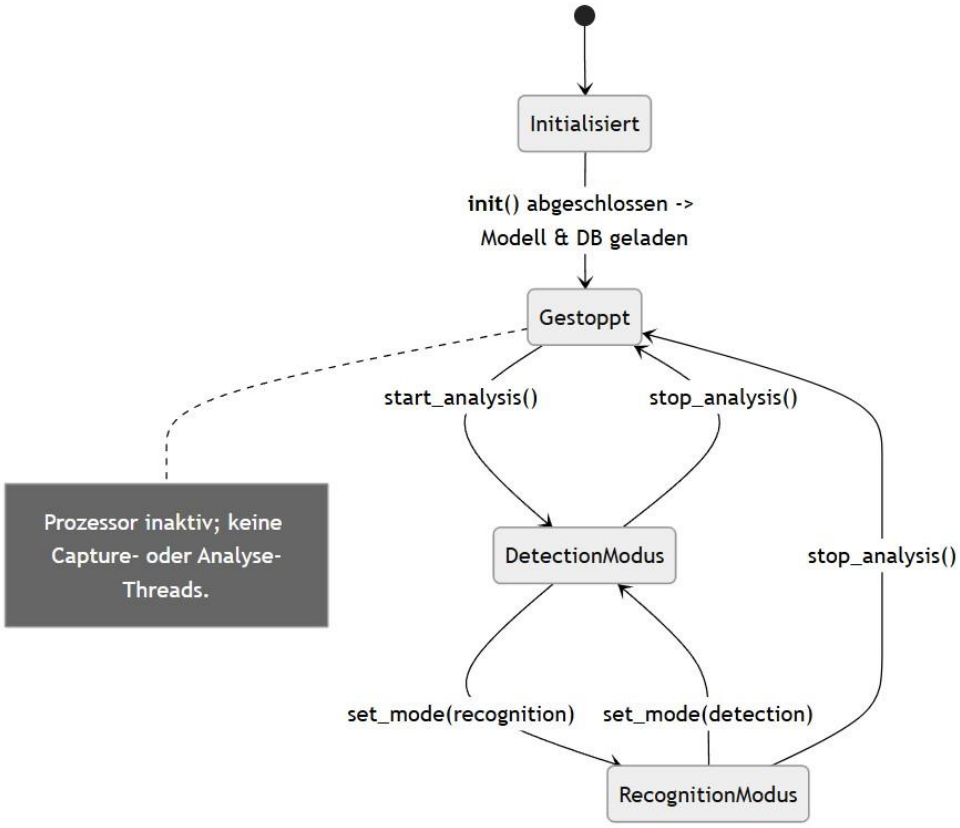


Figure 3.14: *Methoden Gesichtsverarbeitung für Start und Ende im laufenden Programm*

Innerhalb des übergeordneten Läuft-Zustands kann das System dynamisch zwischen den beiden Unterzuständen wechseln. Der Übergang zwischen dem Detection-Modus und dem Recognition-Modus wird durch den Aufruf der Methode set_mode() gesteuert. Diese gezielte

Modusumschaltung ist ein wesentliches Merkmal für die Flexibilität der Klasse. Von besonderer Bedeutung sind hierbei die sogenannten Guard-Bedingungen, die die Gültigkeit von Zustandsübergängen sicherstellen. So ist der Wechsel in den Recognition-Modus oder der Start des Systems direkt in diesem Modus an die Bedingung geknüpft, dass die Gesichtsdatenbank zuvor erfolgreich initialisiert wurde und nicht leer ist. Diese Bedingung verhindert Laufzeitfehler und gewährleistet eine robuste Systemlogik, die im Diagramm explizit visualisiert werden kann. Die Beendigung des aktiven Betriebs wird durch den Übergang vom Zustand Läuft zurück in den Zustand Gestoppt dargestellt. Dieses Ereignis wird durch den Aufruf der stop_analysis()-Methode ausgelöst, welche die internen Threads sicher beendet und alle aktiven Prozesse anhält. Das System kehrt damit in seinen stabilen Ruhezustand zurück und ist bereit für einen erneuten Start.

3.7 Ablaufsteuerung und Exekution

Für ein konsistentes Gesamtsystem war es notwendig, alle zentralen Komponenten detailliert darzustellen. Neben der grundlegenden Browsersteuerung wurden dabei insbesondere die eingesetzten Modelle und architektonischen Bausteine beschrieben. Auffällig war hierbei vor allem die direkte Integration eines Large Language Models als Chatbot in die Klassenstruktur. Darüber hinaus spielte auch die Analyse der eingesetzten Algorithmen eine wesentliche Rolle. Dazu zählen neben den Verfahren der Sprachverarbeitung – etwa für Transkription und Sprachsynthese – auch die Methoden der Bildverarbeitung, die im Rahmen der Gesichtserkennung genutzt werden. Das gesamte System ist logisch miteinander verknüpft und basiert auf einer klaren Ordner- und Klassenstruktur, die ein effizientes Zusammenspiel der einzelnen Module ermöglicht.
Zur Unterstützung zukünftiger Anwender wurde eine ausführliche Projektdokumentation im Markdown-Format erstellt und in einer README-Datei festgehalten. Diese Dokumentation beschreibt nicht nur die Funktionsweise des Systems, sondern listet auch die genutzten Bibliotheken und deren jeweilige Versionen auf. Dadurch wird sichergestellt, dass das Projekt auf unterschiedlichen Systemen reproduzierbar lauffähig bleibt. Gleichzeitig trägt diese Transparenz zur besseren Nachvollziehbarkeit der technischen Anforderungen und der verwendeten virtuellen Umgebung bei.
Neben der funktionalen Implementierung mussten auch Aspekte der Sicherheit berücksichtigt werden. Besonders relevant ist hierbei der Umgang mit API-Schlüsseln. Diese sollten nicht direkt im Quellcode hinterlegt, sondern als Umgebungsvariablen eingebunden werden. Auf diese Weise wird die Gefahr eines Missbrauchs, beispielsweise durch den Diebstahl von API-Credits, erheblich reduziert.
Ein weiterer kritischer Punkt betrifft die Verarbeitung personenbezogener Daten. Durch die speichereffiziente Architektur werden zwar während des Betriebs keine Audioaufnahmen oder Bilder dauerhaft gespeichert. Dennoch ist es notwendig, vorab Referenzbilder in einer Datenbank abzulegen, um die Gesichtserkennung durchführen zu können. Diese Daten stellen potenziell ein Sicherheitsrisiko dar. Sie sollten daher bei der Versionsverwaltung, beispielsweise über entsprechende .gitignore-Einträge, bewusst ausgeschlossen werden. Ergänzend ist es ratsam, klare Richtlinien für den Umgang mit diesen Daten zu definieren, um sowohl datenschutzrechtliche Anforderungen als auch ethische Aspekte zu berücksichtigen.

Kapitel 4

Evaluation und Ergebnisse

Mit den im vorherigen Kapitel erwähnten Modulen konnte geklärt werden, welche Architektur für ein in sich logisch laufendes System notwendig ist. Dabei wurde jedoch nicht weiter darauf eingegangen, inwiefern die Wahl der genannten Modelle stattfand. Ein grundlegendes Ziel war neben der Performance auch die Latenzminimierung. Dementsprechend muss neben modellspezifischen KPIs immer ein Blick auf die Geschwindigkeit der Exekution geworfen werden. Diese wird im Folgenden jederzeit mit betrachtet und ist unter anderem ein ausschlaggebender Faktor, um echtzeitähnliche Performance zu erhalten. Darüber hinaus werden Ergebnisse anhand von echten Beispielen validiert, um praxisrelevant implementiert zu werden.

4.1 Evaluationsmethodik

Für einheitliche Ergebnisse über alle Metriken und Modelle hinweg ist eine klare Dokumentation der zur Verfügung stehenden Ressourcen notwendig. Wie in vorherigen Kapiteln bereits erwähnt, befindet sich im Hardware-Setup neben einer AMD Ryzen 7 9700X auch eine Nvidia 5700 Ti, um Rechenoperationen und für CUDA optimierte Deep-Learning-Frameworks reibungslos ablaufen zu lassen. Für die Evaluierung in praxisrelevanten Umgebungen werden neben dem Stream der Meta Glasses auch Videos aus Interviews und von YouTube entnommen. Damit beim Testen keine technischen Limitierungen durch Internetverbindung und Geschwindigkeit entstehen, wurde diese ebenfalls initial getestet und beobachtet. Diese liegt im Download bei 400 Mbit/s und im Upload bei 12.33 Mbit/s. Dies sorgt für geringe Latenz in der direkten Kommunikation mit den Google-Servern.

4.1.1 Datensätze

Für die konkrete Evaluierung von Modellen und KPIs gibt es branchenspezifische Datensätze, die oftmals vorgefertigt und aufbereitet wurden. Diese wurden primär eingesetzt, um in mehreren Testläufen valide Ergebnisse zu erzielen.

WIDER FACE

Der WIDER FACE Datensatz ist ein Datensatz, der umfassend für die Gesichtserkennung im Bereich der Bildverarbeitung und Computer-Vision eingesetzt wird [191]. Die Besonderheit gegenüber anderen Datensätzen liegt vor allem in seiner Größe und Variabilität, was ihn ideal für das Training von Modellen macht. Dabei sind neben Variationen in Skalierung, Posen und Verdeckungen auch verschiedenste Gesichtsausdrücke, Beleuchtungen und Schärfen zu finden. Angelernte Systeme meistern mit diesen Herausforderungen ein starkes Potenzial zur Verallgemeinerung in unbekannten Verhältnissen. Mit knapp 32.203 Bildern, in denen etwa 394.000 Gesichter annotiert wurden, gibt es eine unzählige Menge an Beispielen. Dabei werden zu jedem Gesicht die Bounding Boxes mit Position und Größe des Gesichts dokumentiert. Daneben wird zu jeder Bounding Box angegeben, inwiefern sich die zuvor genannten Attribute verhalten. Mit dieser Fülle an Informationen ist dieser Datensatz ideal für die später zu sehenden Benchmark-Tests rund um die Gesichtserkennung geeignet.



Figure 4.1: *Beispiele aus dem Wider Face Datensatz mit einer Vielfalt an Bildern[191]*

Labeled Faces in the Wild

Im Gegensatz zur Gesichtsdetektion reicht bei der Erkennung und Identifikation nicht nur die Frage öb", sondern auch "wer". Die Kunst, neben der Detektion Gesichter gleichzeitig zuordnen zu können, findet im Alltag immer öfter Gebrauch. Doch die einwandfreie Durchführung erfordert Modelle, die dementsprechend trainiert wurden. Dies geschah unter anderem mittels des Labeled Faces in the Wild (LFW) Datensatzes [192]. Dabei handelt es sich um eine Bilddatenbank von Gesichtern, die speziell im Zuge der Gesichtsforschung entwickelt wurde. Mit seinem Fokus auf realitätsnahe Aufnahmen wurden neue Maßstäbe gesetzt, was die Anwendbarkeit von Modellen nach dem Training anging. Auch hier lassen sich wieder Unterschiede in Posen, Blickwinkeln, Identitäten, Hintergründen, Okklusionen und weiteren Attributen finden. Mit Ursprung aus Onlinequellen konnte so leicht Vielfalt in die Bilder eingeführt werden. Mit einer Größe von 13.000 Bildern, in denen 5.749 Personen zu erkennen sind, gibt es unzählige Möglichkeiten, sein Modell zu trainieren und zu testen. Zusätzlich zu den Originalbildern gibt es verschiedene äusgerichtete"Versionen wie "funneledünd "deep funneled"Bilder. Die "deep funneled"Bilder sollen eine überlegene Leistung für die meisten Gesichtsverifikationsalgorithmen liefern, da sie durch Rotation und Größenänderung so vorverarbeitet wurden, dass

Augen- und Nasenpositionen ähnlicher sind und die Gesichter im Bildzentrum liegen. Der Datensatz enthält Metadaten, die bei der Erstellung von Trainings- und Testsätzen hilfreich sind, beispielsweise Listen von Personen und Paaren für Kreuzvalidierungen. Daher wird für einen Vergleich verschiedener Gesichtsverifikationsalgorithmen der LFW-Datensatz genutzt.



Figure 4.2: *Typische Bilder beim Labeled Faces in the Wild Datensatz [193]*

LibriSpeech

Mit einer Fülle an englischsprachigen Audiodateien ist der LibriSpeech-Datensatz ideal für die Forschung an Automatic Speech Recognition. Voraussetzung dafür ist auch die Zurverfügungstellung geeigneter Transkripte [194]. Die knapp 1000 Stunden Audiomaterial stammen aus LibriVox-Hörbüchern, die rechtlich einwandfrei genutzt werden können und von einer Vielzahl an Sprachen und Stimmen profitieren. Neben der Varianz ist die Qualität im Allgemeinen ideal und ermöglicht es, auf der Grundlage der genauen Texttranskripte auf Satzebene präzise Tests durchzuführen. Der gesamte Datensatz besteht dementsprechend aus WAV-Dateien, passenden Texttranskripten und unterschiedlichen Ausprägungen, die in Form der reinen Qualität der Sprache gemessen werden. Als Standard für Benchmarks wird dieser für den Vergleich bei den Speech-to-Text-Modellen angewendet.

LibriSpeech

Der LJSpeech-Datensatz ist ein öffentlicher Datensatz von Audio-Text-Paaren, der speziell für das Training von Text-to-Speech (TTS)-Modellen entwickelt wurde. Er besteht aus etwa 24 Stunden gesprochenem Englisch, gelesen von einer einzigen Sprecherin [195]. Die exzellente Qualität der Audiodateien ist ideal für die Erzeugung von synthetischen Stimmen, während Rauschen und andere Störungen minimal sind. Die 24 Stunden Material stammen von einer einzigen Sprecherin und sind damit perfekt für die Erstellung von kohärenten und konsistenten synthetischen Stimmen ohne die Komplexität, die durch mehrere Sprecher eingeführt wird. Der Datensatz besteht aus etwa 13.100 WAV-Dateien, wobei jede Datei einen einzelnen Satz oder eine Phrase enthält. Damit wird die objektive Bewertung von Text-to-Speech-Modellen deutlich erleichtert.

4.1.2 Evaluationsmetriken

Nicht jede KPI lässt sich für jeden Zweck nutzen. Auch wenn Geschwindigkeit im Fokus steht, muss eine Basis geschaffen werden, mit der in der Praxis gearbeitet werden kann. Dazu dienen bei unterschiedlichen Use-Cases spezifische Metriken, deren Ergebnisse Aussagekraft über die genutzten und verglichenen Modelle leisten. Diese wird vorab definiert, erläutert und anschließend mit den Ergebnissen dargestellt und diskutiert.

Precision

Die Precision misst den Anteil an korrekt erkannten positiven Fällen, die unter allen positiv gelabelten Fällen tatsächlich positiv waren. Folgend zeigt dies zum Beispiel, wie viele erkannte Gesichter tatsächlich Gesichter sind. Precision ist besonders wichtig, wenn die Kosten für falsch-positive Ergebnisse (False Positives) hoch sind. In der Gesichtserkennung bedeutet eine hohe False Positive Rate, dass viele Nicht-Gesichter fälschlicherweise als Gesichter erkannt werden, was zu unnötigen Alarmen oder Fehlverarbeitungen führen kann. Ein hohes Precision-Ergebnis zeigt, dass das System, wenn es ein Gesicht erkennt, dies auch mit hoher Wahrscheinlichkeit richtig tut. Es ist jedoch anfällig für ein niedrigeres Recall, das heißt, es könnte viele tatsächliche Gesichter übersehen, um die Präzision zu maximieren.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall

Im Vergleich zur Precision misst der Recall den Anteil der korrekt erkannten positiven Fälle, die unter allen tatsächlich positiven Fällen gefunden wurden, und beantwortet damit die Frage, wie viele tatsächlich vorhandene Gesichter das Modell erkannt hat. Recall ist entscheidend, wenn die Kosten für falsch-negative Ergebnisse (False Negatives) hoch sind. In der Gesichtserkennung bedeutet eine hohe False Negative Rate, dass das System viele vorhandene Gesichter nicht erkennt, was kritisch sein kann, beispielsweise bei Sicherheitssystemen oder der Personenzählung. Ein hohes Recall-Ergebnis zeigt, dass das System sehr gut darin ist, alle tatsächlichen Gesichter zu finden. Es kann jedoch zu einer niedrigeren Precision führen, da das System möglicherweise großzügiger ist und dadurch auch mehr Nicht-Gesichter fälschlicherweise als Gesichter identifiziert.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-Score

Der F1-Score ist das harmonische Mittel aus Precision und Recall. Er bietet ein Gleichgewicht zwischen diesen beiden Metriken und ist besonders nützlich, wenn eine ungleichmäßige Klassenverteilung vorliegt. Der F1-Score ist eine gute Metrik, wenn Sie sowohl Precision als auch Recall gleichermaßen optimieren möchten. Ein hoher F1-Score ist ein Indikator für die Zuverlässigkeit eines Modells, da er eine simultane Reduktion von falsch-positiven und falsch-negativen Klassifizierungen widerspiegelt. Er ist robuster gegenüber Datensatz-Imbalanzen

als die bloße Genauigkeit (Accuracy) und bietet eine einzelne Zahl, die die Leistung eines Modells zusammenfasst, wenn ein Kompromiss zwischen Genauigkeit und Vollständigkeit wichtig ist. Im Kontext der Gesichtserkennung kann ein hoher F1-Score bedeuten, dass das System zuverlässig viele Gesichter findet und dabei wenige Fehler macht.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

mAP

Der mean Average Precision ist eine umfassende Metrik, die häufig zur Bewertung der Leistung von Objekt-Detektionssystemen verwendet wird. Sie ist der Mittelwert der Average Precision über alle Klassen hinweg. Da bei der Gesichtserkennung in der Regel nur eine "Klasse"detektiert wird, entspricht mAP hier der Average Precision der Klasse "Gesicht". Die Average Precision selbst ist eine Zusammenfassung der Precision-Recall-Kurve und wird oft als die Fläche unter dieser Kurve (Area Under Curve, AUC) interpretiert oder als gewichteter Mittelwert der Precision bei verschiedenen Recall-Schwellenwerten berechnet. mAP ist eine sehr robuste Metrik, die sowohl die Lokalisierungsgenauigkeit (mittels Intersection over Union, IoU) als auch die Klassifizierungsgenauigkeit eines Detektors berücksichtigt. Sie fasst die Leistung über verschiedene Konfidenzschwellenwerte und Recall-Niveaus zusammen. Ein hoher mAP-Wert bedeutet, dass das Modell Objekte (Gesichter) mit hoher Präzision und Vollständigkeit über verschiedene Detektionsschwellen hinweg erkennt. Im Gegensatz zu einzelnen Precision- oder Recall-Werten, die oft an einem festen Konfidenzschwellenwert berechnet werden, gibt mAP ein ganzheitlicheres Bild der Systemleistung, da es die gesamte Precision-Recall-Charakteristik berücksichtigt. Es ist die bevorzugte Metrik in vielen Benchmarks für Objekt-Detektion.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

$$\text{mAP (mean Average Precision)} = \text{Mittelwert der Average Precision über alle Klassen}$$

$$\text{AP (Average Precision)} = \text{Fläche unter der Precision-Recall Kurve}$$

Precision-Recall-Kurve

Die Precision-Recall-Kurve ist eine grafische Darstellung des Kompromisses zwischen Precision und Recall bei verschiedenen Konfidenz- oder Detektionsschwellenwerten eines Modells. Für jeden möglichen Schwellenwert wird ein Punkt in einem 2D-Diagramm geplottet, wobei Recall auf der x-Achse und Precision auf der y-Achse liegt. Die PR-Kurve ist ein leistungsstarkes Visualisierungstool, um die Performance eines Detektionsmodells umfassend zu bewerten. Sie zeigt, wie Precision und Recall sich gegenseitig beeinflussen: Wenn Sie versuchen, mehr Objekte zu finden (höherer Recall), sinkt in der Regel die Präzision, da Sie auch mehr Fehler machen. Eine ideale Kurve verläuft nahe der oberen rechten Ecke des Diagramms (hohe Precision bei hohem Recall). Die Form der Kurve gibt Aufschluss über die Robustheit des Modells

gegenüber unterschiedlichen Anwendungsfällen und Schwellenwerten. Sie ist besonders nützlich für unausgeglichene Datensätze, bei denen die Anzahl der negativen Fälle viel höher ist als die der positiven Fälle . Im Gegensatz zur ROC-Kurve (Receiver Operating Characteristic) ist die PR-Kurve für solche Szenarien aussagekräftiger, da sie sich mehr auf die Leistung der positiven Klasse konzentriert.

Receiver Operating Characteristic Area Under Curve

ROC AUC steht für Ärea Under the Receiver Operating Characteristic Curve"(Fläche unter der ROC-Kurve). Um die Performance eines Klassifikationsmodells über das gesamte Spektrum der Entscheidungsschwellen hinweg zu visualisieren, wird die ROC-Kurve herangezogen. In diesem Diagramm wird das Verhältnis der Sensitivität (True Positive Rate) zur Falsch-Positiv-Rate (FPR) abgebildet, wodurch die Trennschärfe des Modells bei variierenden Schwellenwerten deutlich wird. Die AUC ist die Fläche unter dieser Kurve. Die ROC AUC ist eine weit verbreitete Metrik zur Bewertung von Gesichtserkennungssystemen, da sie eine globale Maßzahl für die Diskriminierungsfähigkeit des Modells über alle möglichen Betriebsbedingungen hinweg liefert. Die True Positive rate misst den Anteil an korrekt erkannten positiven Fällen, während die False Positive Rate den Anteil an falsch erkannten positiven Fällen unter allen tatsächlich negativen Fällen misst.

$$\text{TPR (True Positive Rate)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{FPR (False Positive Rate)} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

Die ROC-Kurve zeigt den Kompromiss zwischen TPR und FPR. Eine ideale Kurve geht durch die obere linke Ecke (TPR = 1, FPR = 0). Die AUC-Messung ist schwellenwertunabhängig und aggregiert die Leistung des Modells. Ein AUC-Wert von 1.0 bedeutet einen perfekten Klassifikator, während ein Wert von 0.5 einem zufälligen Klassifikator entspricht. Ein höherer AUC-Wert deutet auf eine bessere allgemeine Leistung des Gesichtserkennungssystems hin, da es über eine breitere Palette von Schwellenwerten eine bessere Fähigkeit zur Unterscheidung zwischen autorisierten und unautorisierten Personen aufweist.

Equal Error Rate

Die Equal Error Rate (EER) ist ein Metrikpunkt, bei dem die Fehlerrate der Falschakzeptanz (False Acceptance Rate, FAR) und die Fehlerrate der Falschrückweisung (False Rejection Rate, FRR) gleich sind. Sie wird ermittelt, indem man den Schwellenwert eines Gesichtserkennungssystems so anpasst, dass FAR = FRR ist. Der gemeinsame Wert beider Raten an diesem Punkt ist die EER. Die EER ist eine weit verbreitete Metrik zur Leistungsbewertung von biometrischen Systemen, da sie einen einzelnen, schwellenwertunabhängigen Wert liefert, der die Gesamtleistung des Systems zusammenfasst. Ein niedriger EER-Wert deutet auf ein leistungsstärkeres System hin, das sowohl Falschakzeptanzen als auch Falschrückweisungen minimieren kann. Die false Acceptance Rate ist die Wahrscheinlichkeit, dass das System eine nicht-übereinstimmende Person fälschlicherweise als übereinstimmend akzeptiert. Währenddessen ist die False Rejection Rate die Wahrscheinlichkeit, dass das System eine übereinstimmende Person fälschlicherweise als nicht übereinstimmend akzeptiert.

$$\text{FAR} = \frac{\text{Anzahl der falschen Akzeptanzen}}{\text{Anzahl der Unbefugten, die versucht haben, Zugriff zu erhalten}}$$

$$\text{FRR} = \frac{\text{Anzahl der falschen Rückweisungen}}{\text{Anzahl der autorisierten Personen, die versucht haben, Zugriff zu erhalten}}$$

EER ist der Wert, an dem FAR = FRR ist.

Mit gegebenem Hintergrund:

$$\text{FAR} = \frac{\text{FP}}{\text{FP+TN}} \quad \text{FRR} = \frac{\text{FN}}{\text{FN+TP}}$$

Word Error Rate

Die WER ist die relevanteste Metrik zur Bewertung der Qualität eines Speech-to-Text Systems. Dabei werden Fehler auf Wortebene gemessen, die bei der Transkription entstehen. Als Vergleich dient dazu eine menschliche Referenztranskription. Die Fehler unterteilen sich dabei in Substitutionen, in denen Wörter falsch platziert wurden, Deletionen, in denen Wörter vollständig fehlen, oder Insertionen, bei denen Wörter in der Transkription nicht in der Masterreferenz existieren. Dabei handelt es sich bei der Word Error Rate um eine Metrik, die die Qualität der Umwandlung von Sprache zu Text wiedergibt. Ein niedrigerer Wert ist erstrebenswert. Mit dieser Metrik kann das Verständnis des Textes leichter ermittelt werden. Es muss jedoch erwähnt werden, dass hier semantische Beziehungen nicht berücksichtigt werden, was leicht zu Fehlern führen kann. Das betrifft neben Pluralformen unter anderem auch Zahlen in verschiedenen Schreibweisen.
Mit den Substitutionen S, der Anzahl an Deletionen D, der Anzahl an Insertionen I und der Anzahl an Gesamtwörtern N ergibt sich damit die Formel für die WER:

$$\text{WER} = \frac{S + D + I}{N}$$

Character Error Rate

Die CER fokussiert sich im Gegensatz zur WER auf einzelne Symbole und Zeichen. Das hilft allgemein bei Sprachen ohne klare Wortgrenzen oder wenn eine feinere Analyse der Fehlerursachen erforderlich ist. Auch bei der Nutzung vieler unbekannter Wörter, wie Produktnamen oder Eigennamen, kann das von entscheidendem Vorteil sein. Dementsprechend wird jedoch auch sensibler auf Zeichen- und Tippfehler reagiert. Auch hier ist ein niedrigerer Wert von Vorteil. Mit den gleichen Definitionen nun auf Zeichenebene misst man die CER wie folgt:

$$\text{CER} = \frac{S_c + D_c + I_c}{N_c}$$

Real-Time Factor

Mit dem RTF wurde eine Effizienzmetrik eingeführt, die gerade für dieses Projekt von großer Bedeutung ist. Hierbei wird betrachtet, wie das Verhältnis zwischen der Verarbeitungszeit der Transkription und der tatsächlichen Dauer des Audios aussieht. Für die Echtzeitverarbeitung ist diese Metrik unumgänglich. Ein Wert kleiner als 1 verdeutlicht, dass das System schneller als Echtzeit arbeitet. Damit liegt die Verarbeitungsdauer unter der tatsächlichen Länge der Audiospur. Im gleichen Sinne verhalten sich RTF-Werte gleich oder größer als 1. Diese eignen sich dementsprechend weniger gut für Echtzeit-Dienste und bringen oft deutlich mehr Latenz mit.

$$\text{RTF} = \frac{\text{Gesamte Verarbeitungszeit des Audios}}{\text{Gesamte Dauer des Audios}}$$

Speaking Rate

Die Speaking Rate in "Characters Per Second"(CPS) misst die Geschwindigkeit, mit der ein Sprecher Zeichen pro Sekunde ausspricht. Sie wird berechnet, indem die Gesamtzahl der transkribierten Zeichen (ohne Leerzeichen und Interpunktion) durch die Dauer des gesprochenen Audiomaterials geteilt wird. Als wichtige sprachliche und akustische Metrik gibt sie Aufschluss über den Sprachstil einer Person und Eigenschaften der Datei. Sprechgeschwindigkeiten werden dabei in verschiedenen Kontexten charakterisiert. In einem Text-to-Speech-System, wie es hier vorliegt, kann damit die Natürlichkeit und Verständlichkeit der generierten Sprache bewertet werden. Der Vorteil von CPS gegenüber Words Per Minute (WPM) oder Words Per Second (WPS) ist, dass es sprachunabhängiger ist, da die Definition eines "Wortesïn verschiedenen Sprachen variieren kann.

$$\text{Speaking Rate (CPS)} = \frac{\text{Anzahl der Zeichen (ohne Leerzeichen/Interpunktion)}}{\text{Dauer des Audios in Sekunden}}$$

Bei der Betrachtung und Diskussion der Ergebnisse werden die gezeigten Metriken als Vergleich herangezogen und kritisch hinsichtlich Latenz sowie Praxisimplementierung hinterleuchtet.

4.2 Ergebnisse der Teilkomponenten

4.2.1 LLM

Im Rahmen dieser Untersuchung wurde ausschließlich das Large Language Model von Gemini über dessen API getestet. Diese Fokussierung ermöglichte eine konsistente und standardisierte Datenerhebung, die für die Vergleichbarkeit der Ergebnisse von entscheidender Bedeutung war. Die Entscheidung, sich auf ein einziges Modell zu konzentrieren, basierte auf der Annahme, dass die grundlegenden Leistungsmerkmale anderer vergleichbarer LLMs in diesem spezifischen Kontext keine signifikanten Abweichungen aufweisen würden, die die Validität der Studienergebnisse beeinträchtigt hätten. Vorstudien und die aktuelle Forschung legen nahe, dass viele moderne LLMs in ihrer Kernfunktionalität und ihrer Fähigkeit, bestimmte Aufgaben zu lösen, eine hohe Konvergenz aufweisen. Daher wird angenommen, dass die hier

gewonnenen Erkenntnisse in einem größeren Kontext von LLM-Anwendungen verallgemeinerbar sind [196, 197]. Die zur Verfügung gestellten Modelle reichen von Geminis altem Gemini 1.5 Flash bis hin zu den neuesten State-of-the-Art-Modellen wie Gemini 2.5 Pro oder auch der deutlich kleineren Variante Gemini 2.5 Flash lite [48]. Mit einem besonderen Fokus werden auch nur diese in verschiedenen Aspekten betrachtet.
Bevor der limitierende Faktor Zeit näher betrachtet wird, soll ein Blick auf die Antwortlängen geworfen werden. Dazu wurden fünf Beispielfragen gestellt, die als Maßstab konsistent über alle Modelle dienen sollten. Als Anweisung kam der Befehl, diese Fragestellungen präzise zu erklären.

```
Explain quantum computing in simple terms.
What are the main challenges in developing artificial general intelligence?
Summarize the plot of 'The Lord of the Rings' in 100 words or less.
Write a short, creative haiku about a sunset.
What is the capital of France and why is it famous?
```

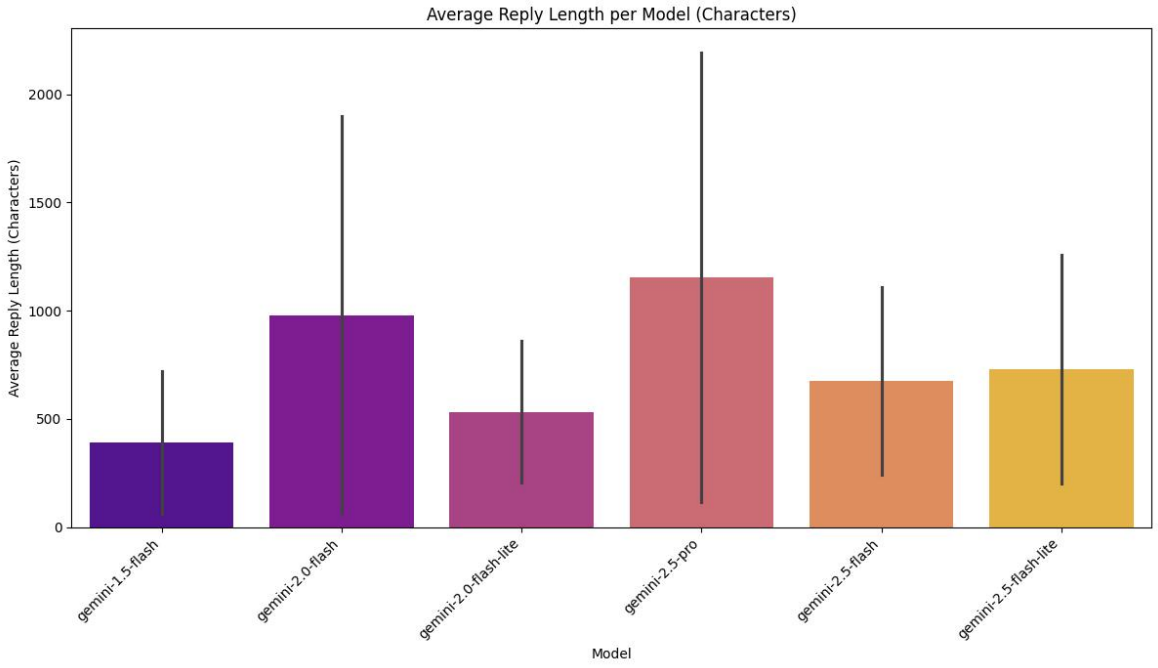


Figure 4.3: *Durchschnittliche Antwortlänge verschiedener Gemini-Modelle in Zeichen*

Mit einem Blick auf die durchschnittlichen Antwortlängen werden erste Unterschiede klar. Das Flagship-Modell 2.5 Pro gibt mit Abstand die längsten Antworten. Das ist vor allem auf seine Architektur und Instruktion zurückzuführen, die mit transparenten Gedankengängen komplexe Aufgaben lösen sollen und daher eher dazu neigt, ausführlicher zu antworten. Mit Blick auf Latenz und flüssige Konversationen muss dies jedoch zunächst neutral bewertet werden. Die leichteren Modelle zeigen, wie vermutet, kürzere Antworten, unterscheiden sich

untereinander jedoch noch deutlich. Gerade der Sprung in der durchschnittlichen Antwortlänge zwischen Flash 2.0 und 2.5 ist bemerkenswert. Die kürzesten Antworten liefert hierbei das älteste Modell von Google. Erst mit einem Blick auf die einzelnen Antworten können Schlüsse über wirklich passende Antwortlängen gezogen werden.

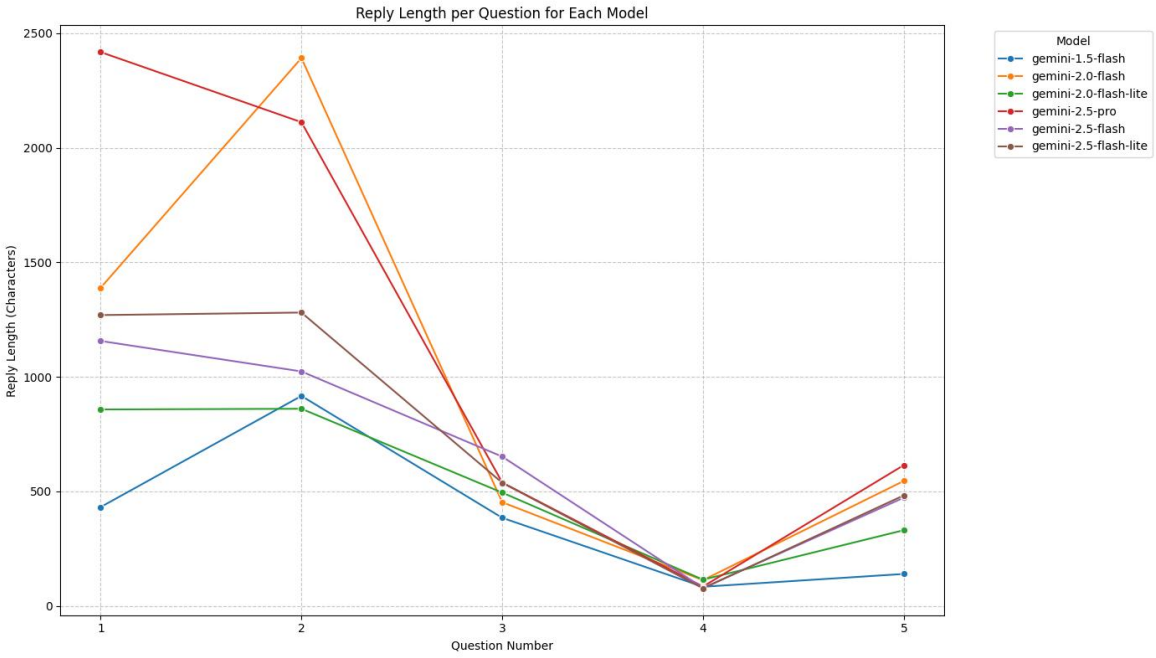


Figure 4.4: *Antwortlänge der Gemini-Modelle pro Testfrage*

Abbildung 4.4 verdeutlicht anhand der gestellten Fragen die Vor- und Nachteile der neueren Modelle. Während komplexe Antworten deutlich längere Zeit benötigen, wird bei simplen Aufgaben wie Zusammenfassungen die Variabilität verkleinert und erreicht bei konkreten Anweisungen "Write a short, creative haiku about a sunset.ëin minimales Niveau. Nach den Antwortlängen ist nun die fundamentalste Komponente dran: die Dauer der zeitlichen Verarbeitung.

Erst hier werden die signifikanten Unterschiede deutlich, wie in Abbildung 4.5 ersichtlich ist. Mit einer durchschnittlichen Antwortzeit von 15,56 Sekunden liegt Gemini 2.5 Pro weit über allen anderen. Dies muss vorerst nicht negativ sein, führt aber unnötigerweise zu großen Verzögerungen, die in realen Konversationen unnatürlich wirken und neben der Sprachverarbeitung nur als zeitlicher Ballast dienen. Dahingegen beweist sich das 2.5 Flash-Lite-Modell als weitaus schneller. Mit einer durchschnittlichen Antwortzeit von 1,1 Sekunden ist es damit das schnellste aller ausgewählten Modelle. Eine nahezu durchgehende Dominanz zeigt sich auch auf Ebene der einzelnen Anweisungen. Während Gemini 2.5 Pro die detailliertesten Antworten liefert, wird dementsprechend bei jeder Frage Zeit in Anspruch genommen. Dazu zählt unter anderem auch die Anweisung, die Antwort kurz zu halten. Trotz der wenigen Zeichen in Antwort vier beträgt hier die Antwortzeit einen Maximalwert. Dahingegen schaffen es die Lite-Modelle durchgehend, schnell zu bleiben und teilen sich über die Anweisungen hinweg die minimale Latenzzeit, wie in Abbildung 4.6 zu beobachten ist.

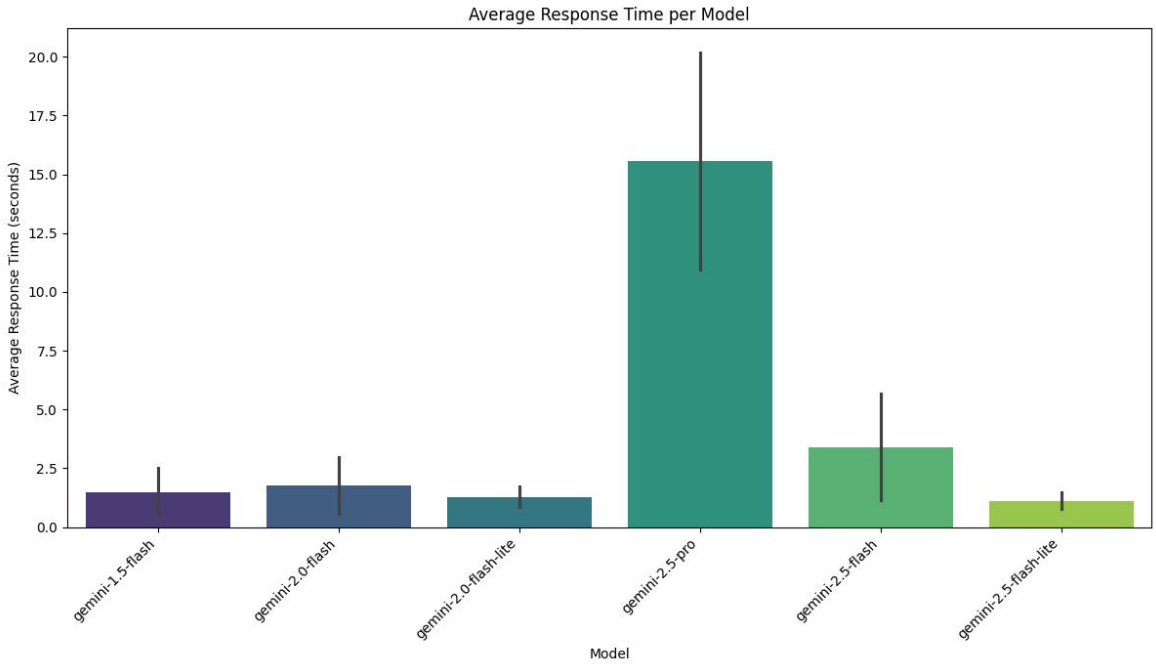


Figure 4.5: *Durchschnittliche Antwortzeit verschiedener Gemini-Modelle in Sekunden*

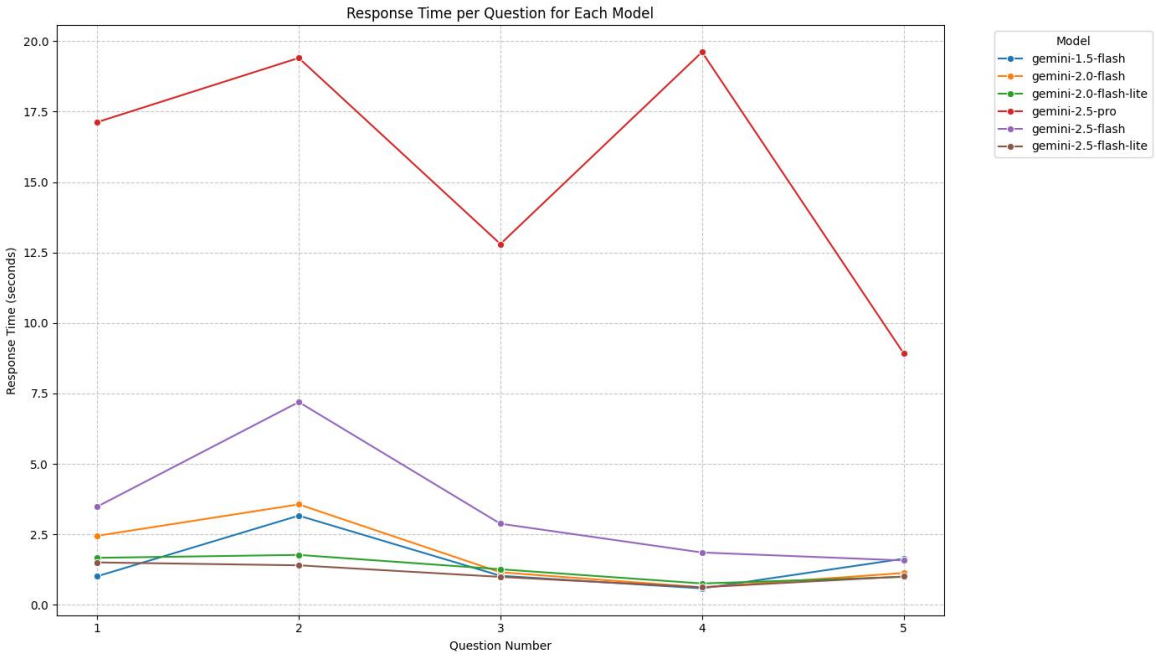


Figure 4.6: *Antwortzeit der Gemini-Modelle pro Testfrage*

Dies führt zur Schlussfolgerung, dass Geminis 2.5 Flash-Lite nicht nur aufgrund seiner Antwortlängen, sondern vor allem auch aufgrund seiner Kapazität, Neuheit und State-of-the-Art-Architektur mit multimodalen Fähigkeiten und vor allem Latenz zur Implementierung herangezogen wird.

4.2.2 Speech-to-Text

Für den Automatic Speech Recognition Part sollen unterschiedliche Perspektiven in Erwägung gezogen werden. Dabei werden nicht nur unterschiedliche Modellgrößen (Tiny bis Large) und Modelle (Whisper, Faster Whisper, Turbo, Vosk) getestet, sondern auch beobachtet, wie sich dies im Vergleich zur Cloud-API-Nutzung verhält. Dieser Schritt wurde gerade bei den LLM's vollzogen, da moderne Sprachmodelle enorme Rechenleistung erfordern. Neben den zuvor definierten Metriken WER, CER und RTF soll für spezielle Fälle auch berücksichtigt werden, wie sich die Transkriptionszeit im Verhältnis zur Länge der Audiospur verhält. Für das Benchmarking wurden für nahezu alle Modelle 100 Audiodateien genutzt. Nur für die Inferenz auf Groq wurden aufgrund von API-Limitierungen nur 20 Dateien genutzt.

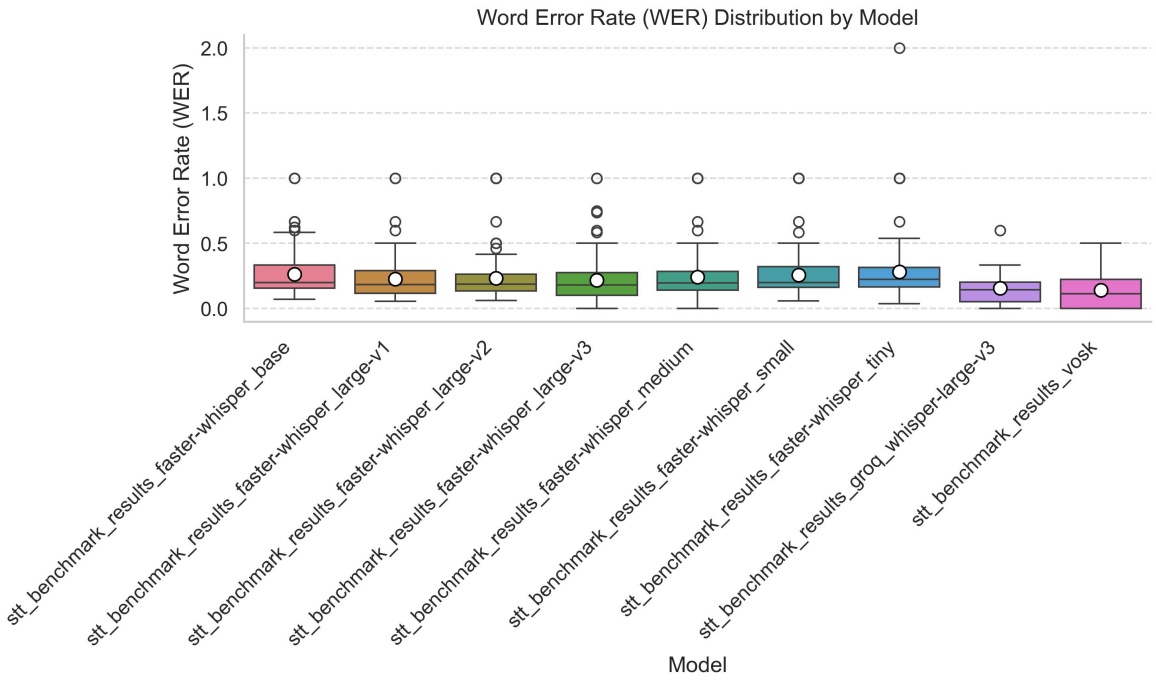


Figure 4.7: *Verteilung der Wortfehlerrate für verschiedene Speech-to-Text-Modelle*

Aus der Übersicht in Abbildung 4.7 lässt sich ableiten, dass das cloudbasierte Modell zusammen mit Vosk niedrigere Word- und Character-Error-Rates erzielt. Diese unterbieten sogar die Faster Whispers Large-Modelle. Diese Ergebnisse weisen im ersten Moment auf die Nutzung der Letzteren hin, da sie sich aufgrund ihrer Qualität der gegebenen Ergebnisse deutlich hervorheben. In Tabelle **??** werden die Resultate nochmals verdeutlicht. Spannend ist dabei zu beobachten, dass die Fast Whispers Medium- und Large V3 Varianten kaum auseinanderliegen, was sich als Vorteil in den Geschwindigkeiten erweisen kann.

Model	mean WER	mean CER
FW (Fast-Whisper) Tiny	0.281	0.088
FW Small	0.257	0.075
FW Base	0.263	0.082
FW Medium	0.240	0.069
FW Large V1	0.225	0.056
FW Large V2	0.223	0.061
FW Large V3	0.217	0.067
Groq Whisper turbo Large	0.155	0.045
Vosk	0.142	0.044

Table 4.1: *WER und CER alle getesten Modell*

Spannend wird es bei einem Blick auf den Realtime Factor, der, wie zuvor gezeigt, einen Eindruck über die zeitliche Performance gibt. Mit Regressionslinien zur Hilfestellung wird klar deutlich, dass trotz der guten Performance Groqs Whisper Modell deutlich und VOSK leicht über den Varianten von Fast Whisper liegen. Diese Ergebnisse bestätigen die kontinuierlichen Fortschritte in der Optimierung rund um die Exekution auf Endgeräten. Erst ab einer Länge von 25 Sekunden zeigen sich Vorteile in der Inferenznutzung von Groq, was sich jedoch für normale Konversationen und Befehle erübrigt. Mit einem durchschnittlichen RTF von 0.032 ist die Medium-Variante um 0.104 besser als VOSK und um 0.085 besser als Groqs Whisper Turbo. Auf der anderen Seite ist jedoch zu erwähnen, dass, wie zu erwarten, die Tiny-Variante mit einem RTF-Wert von 0.021 der klare Sieger ist.

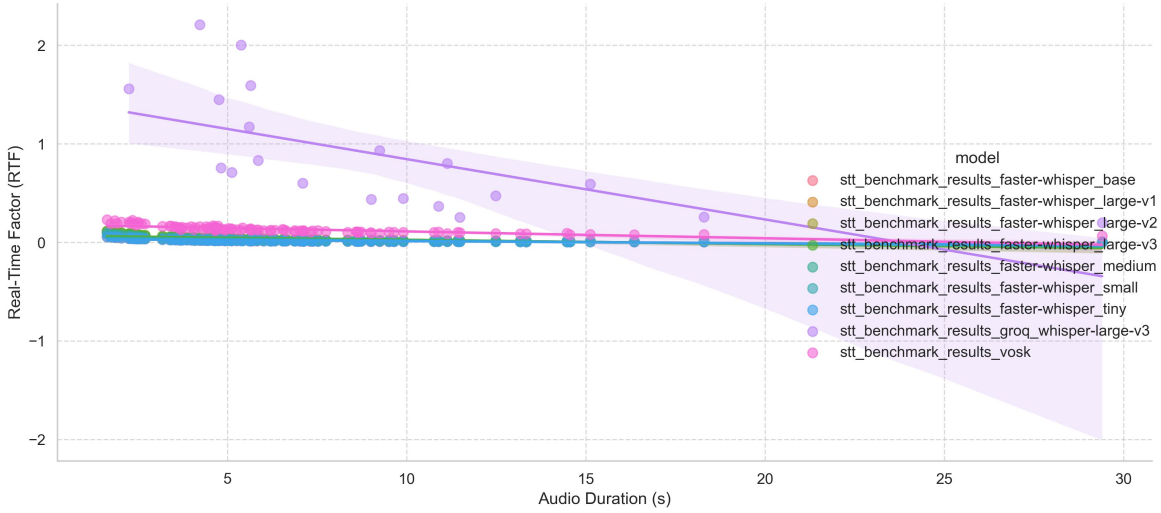


Figure 4.8: *Real-Time-Factor (RTF) der STT-Modelle in Abhängigkeit von der Audiodauer*

Ähnliches Verhalten lässt sich in der letzten Übersicht zur Transkriptionszeit feststellen. Dabei wurde Letztere gegen die Dauer der Audiospur gelegt, um zu zeigen, wie sich die Modelle mit fortlaufenden Audiolängen verhalten. Um ein System zu skalieren und robust zu

fertigen, sind Verläufe anzustreben, die kaum positiv und optimalerweise negativ steigen.

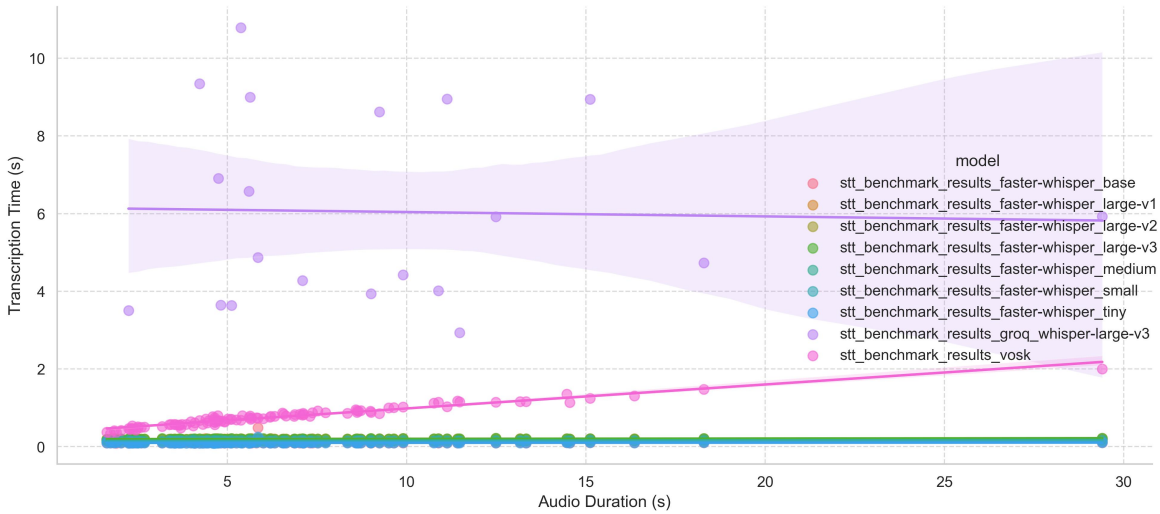


Figure 4.9: *Transkriptionszeit der STT-Modelle in Abhängigkeit von der Audiodauer*

Abbildung 4.9 gibt eine Indikation darüber, wie die Werte in der Praxis aussehen. Es fällt direkt auf, dass Groq und VOSK deutlich zunehmen und nicht konstant sind. Dies kann sowohl auf Inferenz und API-Requests als auch auf grundlegende Architekturen zurückgeführt werden.
Abschließend lässt sich sagen, dass trotz positiver Eigenschaften bezüglich Word und der Character Error Rate Groqs Whisper Large Turbo und VOSKs Modell in der Skalierung deutliche Schwächen zeigen, während Fast Whispers Modell enorme Vorteile in der Geschwindigkeit als Trade-off gegenüber den Fehlern bietet. Mit einer Transkriptionszeit von durchschnittlich 0,152 Sekunden gegenüber den 0,197 des großen Modells und den 0,098 Sekunden des Tiny-Modells bietet die Medium-Variante einen idealen Ausgleich zwischen Performance und Geschwindigkeit, der für das vorliegende Projekt von Bedeutung ist. Daher wird sich für dieses Modell entschieden.

4.2.3 Text-to-Speech

Neben der Spracherkennung und der Textgenerierung ist im letzten Schritt der Sprachverarbeitungs-Pipeline die Sprachsynthese von enormer Bedeutung. Dabei ist jedoch nicht nur eine reine Betrachtung der Metriken wichtig, sondern auch die qualitative Bewertung gegebener Stimmen. Für die Tests wurden zehn zufällige Dateien aus dem LibriSpeech-Datensatz herangezogen.
Ein erster Blick wird dabei auf die Geschwindigkeit geworfen, mit der die Stimmen in den Audiodateien erzeugt werden. Enorme Vorteile erweisen sich in den Architekturen von FastPitch und Pyttsx3, die mit ihrer Geschwindigkeit um ein Vielfaches kürzere Laufzeiten benötigen als die vergleichbaren Modelle. Auffällig ist hierbei ebenfalls der RTF des Sesame Conversational Speech Models. So erweist sich die Fähigkeit, synthetisch gesprochene Wörter kontextabhängig zu vertonen, eher als rechenintensive Übung denn als leistungsgetriebener Vorteil. Im Gegensatz dazu erzielt FastPitch trotz seiner Fähigkeiten erstaunliche Geschwindigkeiten. Trotz

der Tatsache, dass Output Tokens von Gemini 2.5 Flash-Lite begrenzt werden können, muss Rücksicht auf längere Textformen genommen werden. Abbildung 4.11 schafft ein klares Bild, wenn die Frage nach Skalierbarkeit aufkommt. Für längere Antworten beginnen die Modelle, sich deutlich zu unterscheiden. Das fällt nicht nur mit einem Blick auf Sesame auf, sondern beispielsweise auch bei Tacotron, der mit ansteigender Tokenzahl immer länger für die Synthese benötigt. Erstaunlicherweise halten sich FastPitch und Pyttsx3 nicht nur konstant in niedrigen Wertbereichen auf, sondern zeigen auch eine klare Tendenz bei der Begutachtung der benötigten Latenz im System.

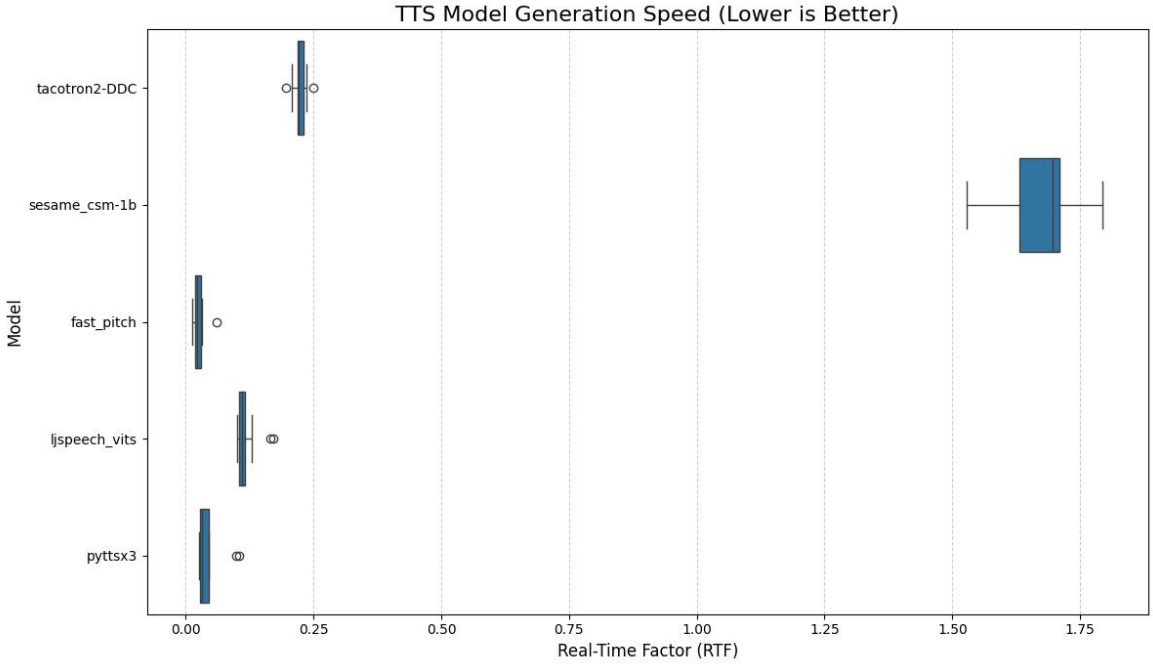


Figure 4.10: *Generierungsgeschwindigkeit von Text-to-Speech (TTS)-Modellen, gemessen als Real-Time-Factor (RTF)*

Mit den gegebenen Informationen und Grafiken ist die Frage nun nur noch, ob sich FastPitch oder Pyttsx3 mehr lohnt und welches in der Praxis deutlicher greifbar ist. Wie zu Beginn erwähnt, sind neben der Geschwindigkeit auch die Tonlage und Stimme entscheidende Faktoren. Beim Erstellen von Audiodateien wird demnach ohne Probleme ein klarer Vorteil von FastPitch erkannt. Mit seiner Fähigkeit, kontextabhängige Sprache zu generieren, die nicht nur natürlich und authentisch klingt, sondern auch schnell erzeugt wird, erweist sich das Modell als klarer Favorit gegenüber der betriebssystemintern arbeitenden Pyttsx3-Architektur. Dies spiegelt sich nicht nur in der quantitativen, sondern auch in der qualitativen Betrachtung anhand der Speaking Rate wider. Aufgrund dieser Tatsachen fällt die Wahl für die Praxis leicht.

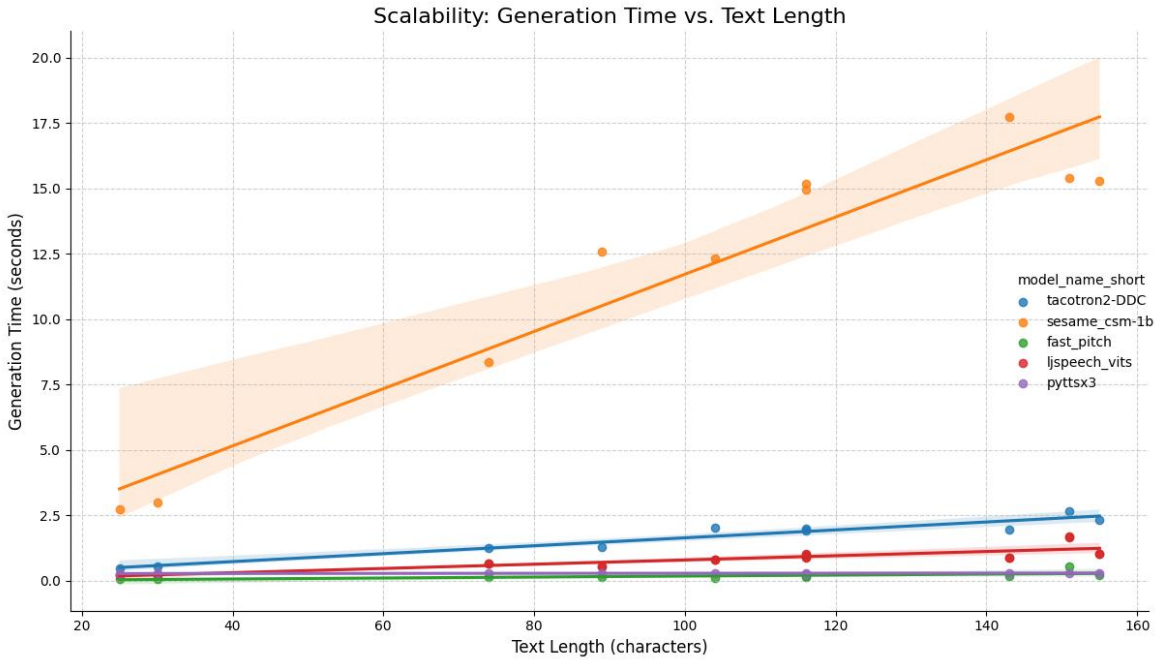


Figure 4.11: *Skalierbarkeit der TTS-Modelle: Generierungszeit in Abhängigkeit von der Textlänge*

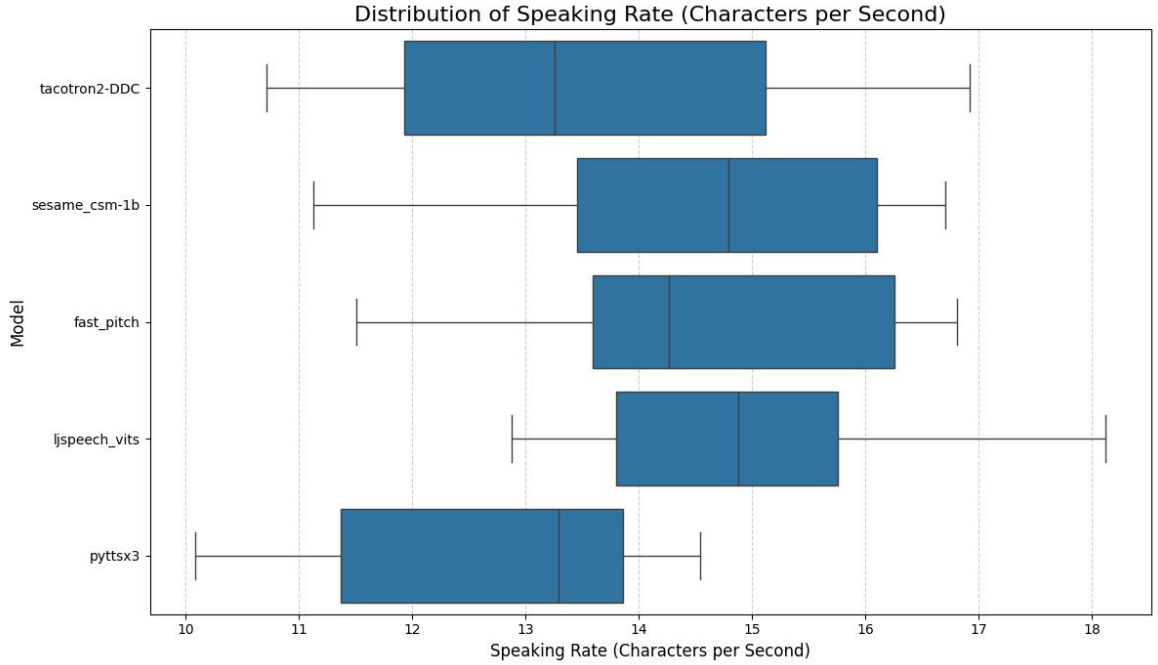


Figure 4.12: *Verteilung der Sprechgeschwindigkeit (Zeichen pro Sekunde) für verschiedene TTS-Modelle*

4.2.4 Face Detection

Als eine der wichtigsten Komponenten ist ohne Zweifel die Fähigkeit, Gesichter in kurzer Zeit mit maximaler Zuverlässigkeit zu erkennen. Dafür werden einige Modelle und Architekturen anhand des Wider-Face-Datensatzes verglichen, der sich durch seine Fülle an Bildern und zugehörigen Metadaten idealerweise eignet. Für diesen Zweck werden jedes Mal 500 Beispielbilder in Betracht gezogen und anhand dieser bewertet. Die Hemmschwelle für die Detektion und Anerkennung wird hierbei auf 0,5 festgelegt. Alles ab diesem Wert wird gelabelt und mitgewertet, der Rest wird ignoriert. Unter den neuen SOTA-Modellen werden unter anderem MediaPipe, DNN SSD, YuNet, Yolos Pretrained v8-Face und Pretrained ArcFace-Modelle in zwei unterschiedlichen Variationen in Erwägung gezogen. Auf der gegenüberliegenden Seite stehen alte "mechanische"Prozesse mit Haar Cascade, LBP Cascade und DLIB HOG zur Verfügung.

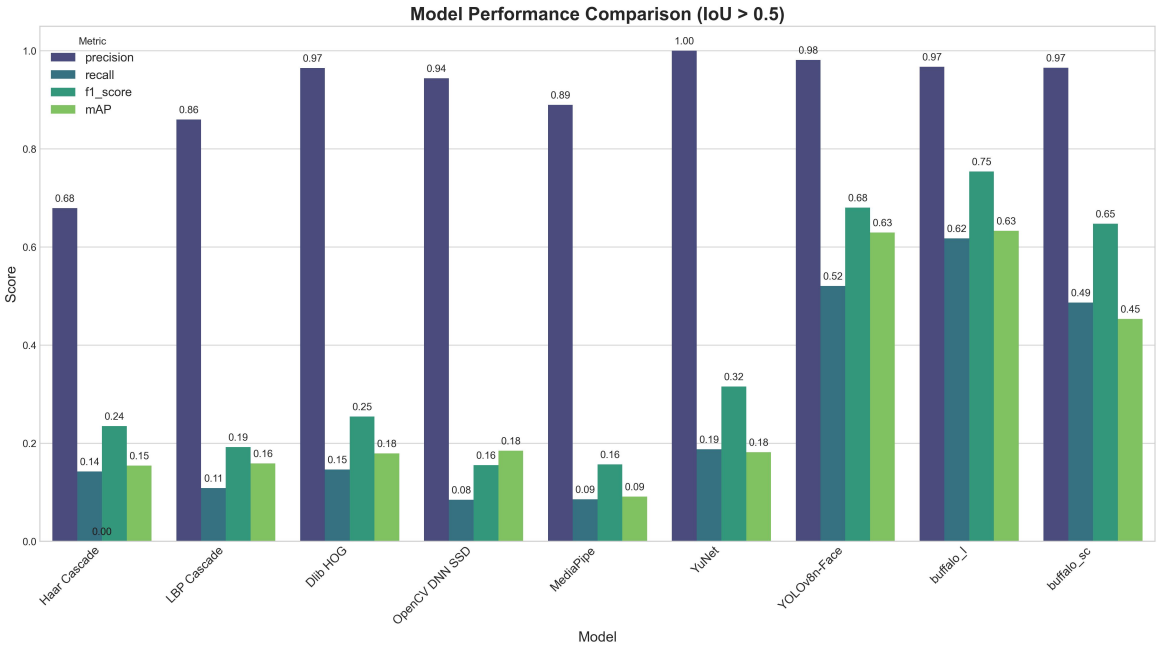


Figure 4.13: *Leistungsvergleich von Gesichtsdetektionsmodellen anhand von Precision, Recall und F1-Score*

Eine erste Übersicht weist bereits auf starke Unterschiede in verschiedenen Metriken hin. Cascade-Modelle erweisen sich in der Präzision als weitaus schlechteste Wahl. Sie können, von links nach rechts blickend, nicht mit neueren Methoden mithalten, was sich auch in den konkreten Benchmark-Werten widerspiegelt. Obwohl in 4.13 die Präzision bei SSD, MediaPipe und YuNet deutlich besser bewertet wurde, hängen diese bei allen anderen Metriken durchaus hinterher. Mit klarem Abstand beweist sich das für Gesichter feingetunte Yolo-Modell mit beiden feingetunten ArcFace-Modellen als klarer Sieger über alle Metriken hinweg. Leichte Unterschiede zwischen den beiden ArcFace-Variationen Buffalo Large und Small sind zwar zu erkennen, doch sind sie auf die Modellgröße zurückzuführen. Rein aus den quantitativen Ergebnissen würde sich hier bereits eine klare Wahl treffen, doch weitere Metriken sind ebenfalls entscheidend. Ein deutlicher Trade-off wird in der nachfolgenden Grafik ersichtlich.

Auch wenn MediaPipe das mitunter schlechteste Modell ist, was Recall, F1 und mAP angeht, werden hier in Anbetracht der Inferenz- und Verarbeitungszeit die besten Ergebnisse erzielt. Genau dies wird in der Diskussion über die Wahl eines idealen Modells später noch wichtig.

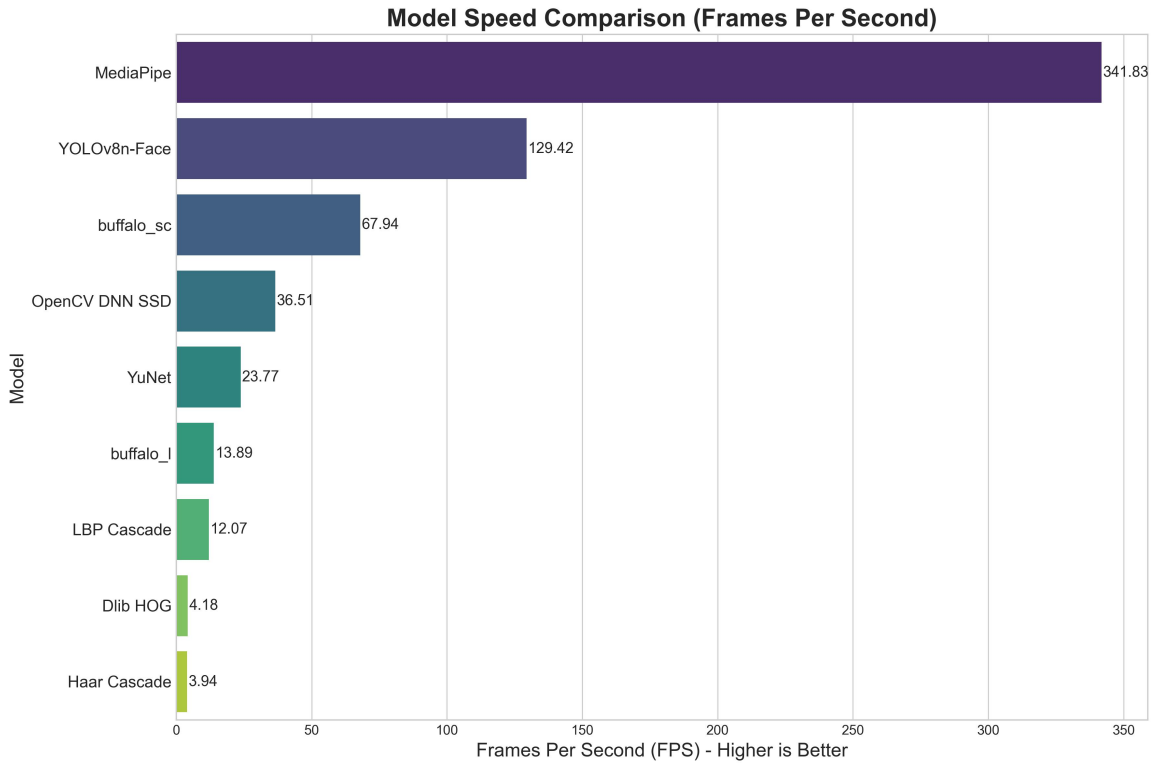


Figure 4.14: *Geschwindigkeitsvergleich von Gesichtsdetektionsmodellen in Frames pro Sekunde (FPS)*

Mit einem Abstand von über 212 und 284 Frames pro Sekunde folgen darauf erst Yolov8n-Face und das kleine Modell der Buffalo-Familie. Auch hier wird nochmals intensiviert, dass ein kleiner Sprung in der Performance und Genauigkeit im großen Buffalo-Modell aus Abbildung 4.13 zu deutlichen Einbußen in der Performance geführt hat, wie ein Vergleich zwischen beiden in Abbildung 4.14 zeigt. Mit einer Verarbeitungsleistung des Large-Buffalo-Modells gegenüber Yolov8 von nur 10,7 Prozent wurde die Geschwindigkeit maßgeblich negativ beeinflusst, was sich in der Praxis wiedererkennen lassen wird. Die Beständigkeit des Trade-Offs wird in einer konkreten Visualisierung in Abbildung 4.15 klar deutlich. Die Grafik ist dabei so zu werten, dass Modelle besser sind, je weiter oben links sie sich befinden. Mit einem deutlichen Vorsprung erweist sich Yolov8n-Face hier als klarer Sieger. Trotz der geringen durchschnittlichen Detektionszeit von MediaPipe, wie zuvor ersichtlich, ist die Mean Average Precision von Yolo nicht zu vernachlässigen, woraufhin das kleine ArcFace-Modell mehr oder weniger knapp folgt. Auffällig ist der hervorragende Wert im Vergleich des Large-Buffalo-Modells gegenüber allen anderen verbliebenen Modellen, die sowohl Schwierigkeiten bei einem guten mAP-Wert als auch bei den durchschnittlichen Detektionszeiten haben. Eine Analyse zwischen Precision

und Recall beweist ein letztes Mal die Dominanz von Yolo und ArcFace. Dabei zeigt sich die Beständigkeit und Robustheit dieser drei ein finales Mal hinsichtlich positiver Klassen in Abbildung 4.16.

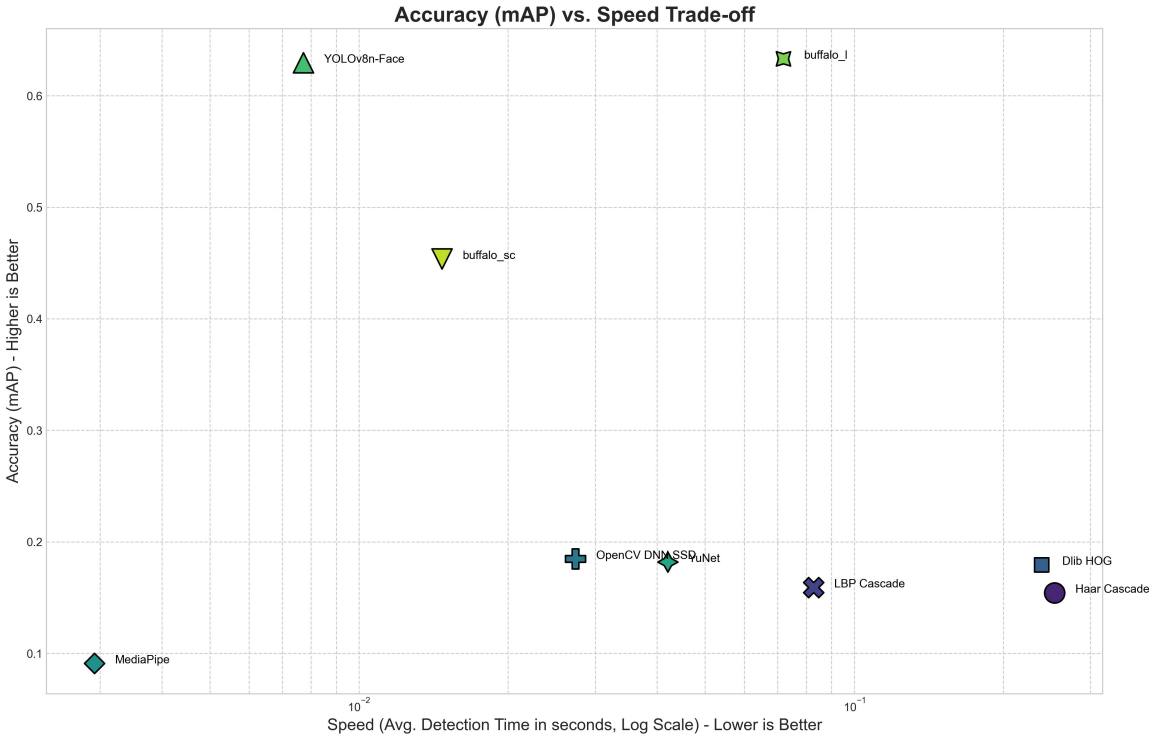


Figure 4.15: *Trade-off-Analyse für Gesichtsdetektoren: Genauigkeit (mAP) vs. Detektionsgeschwindigkeit*

Verständlicherweise stellt sich die Frage, warum unter normalen Umständen mit den zu beobachtenden Werten kein klarer Sieger identifiziert wird. Dies lässt sich mit einem Blick im nächsten Teil klären, in dem des Weiteren Gesichtserkennungsmethoden näher analysiert werden, die im Zusammenspiel mit der Detektion einen entscheidenden Faktor mit sich bringen.

4.2.5 Face Recognition

Im Gegensatz zu den vorherigen Modellen sind das Erkennen und die Identifikation von Gesichtern an eine zweite Bedingung neben der Geschwindigkeit geknüpft. Das Ziel dieser Arbeit ist es, neben einer latenzarmen Inferenz auch sicherzustellen, dass die Bildverarbeitung unter Wahrung der Privatsphäre erfolgt. Das heißt, bei der Erkennung von Gesichtern muss beim Resultat der Analyse sichergestellt werden, dass so wenige Gesichter wie möglich unkenntlich bleiben, die nicht in der Datenbank gespeichert sind. Das heißt, neben der Geschwindigkeit ist die wichtigste Komponente der Recall. Dieser misst den Anteil der tatsächlich unbekannten Gesichter, die von Ihrem Modell korrekt erkannt und unscharf gemacht wurden. Die Benchmarks werden mittels Labeled Faces in the Wild durchgeführt, in denen bis zu 6000 Bilder analysiert werden.

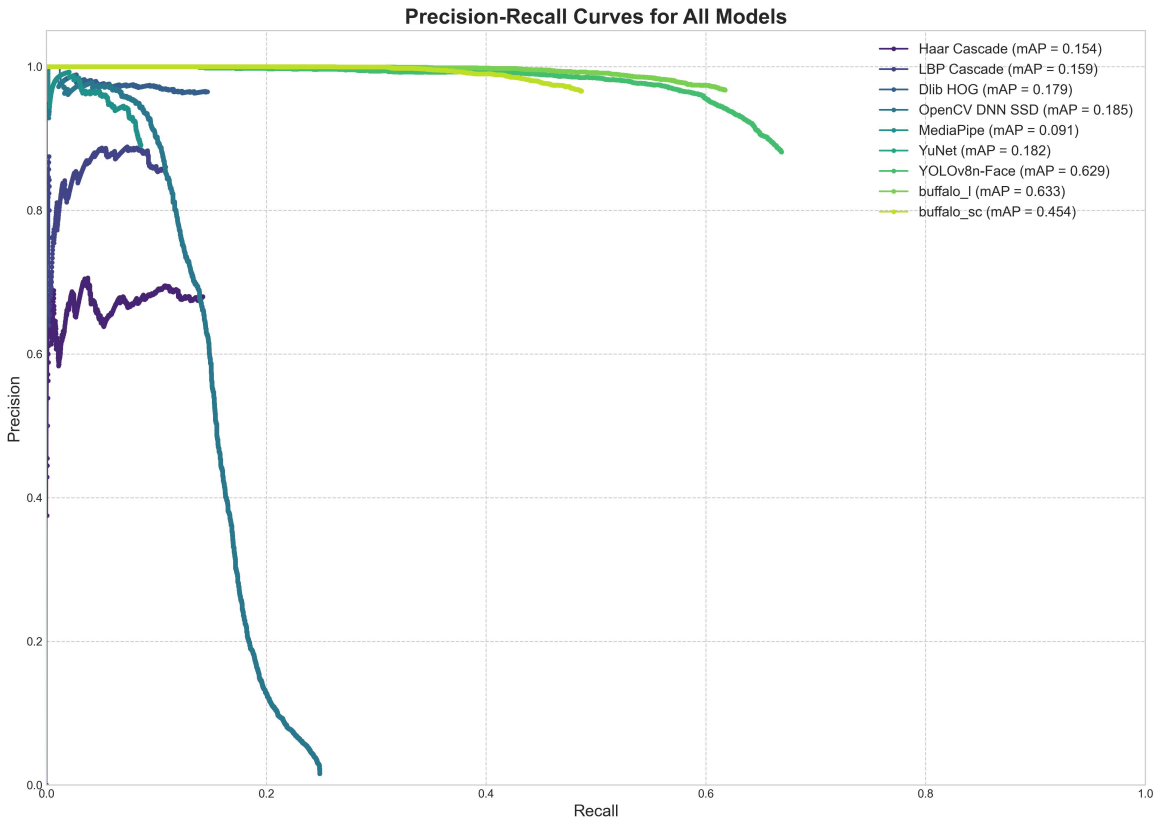


Figure 4.16: *Precision-Recall-Kurven der evaluierten Gesichtsdetektionsmodelle*

Zunächst soll somit ein Blick neben der Präzision auch auf den Recall geworfen werden.

Ohne Schwierigkeiten wird ersichtlich, dass SFace und OpenFace beim Recall zumindest kaum mithalten können. Das liegt vor allem daran, dass es sich bei den beiden nicht um pretrained Modelle handelt, sodass die Differenz umso mehr verdeutlicht wird. Einen klaren Vorsprung haben dabei die ArcFace Modelle buffalo und VGG-Face 2, die sich in beiden Metriken kaum unterscheiden.

Um eine klarere Differenz zu begründen, werden weitere Metriken betrachtet, deren Unterschiede bei der Analyse hilfreich sein sollen. Abbildung 4.18 zeigt erneut, wie ähnlich sich die vier Top-Modelle in den betrachteten KPIs sind. Das zeigt sich nicht nur bei der Accuracy, dem F1-Score und ROC AUC, sondern auch bei der Equal Error Rate. Nun stellt sich die Frage, welches dieser Modelle einen erheblichen Vorteil bietet. Doch eine letzte entscheidende Komponente ist erneut die Zeit, denn diese bestimmt wieder über die Exekution und Praktikabilität bei der Implementierung.

Erstaunlicherweise reiht sich das SC-Modell der ArcFace-Buffalo-Reihe ganz vorne ein. Mit nahezu höchster Genauigkeit und gleichzeitig der schnellsten Laufzeit trennt es sich vom Rest der Modelle und erweist sich als klarer Favorit.

Unter Berücksichtigung der hier gezeigten Ergebnisse soll nochmals auf die Gesichtserkennung geblickt werden. Wie zuvor festgestellt wurde, lagen das Large-Modell und das SC-Modell neben Yolov8n-Face weit voraus, wenn es darum ging, Gesichter zu detektieren. Aufgrund der

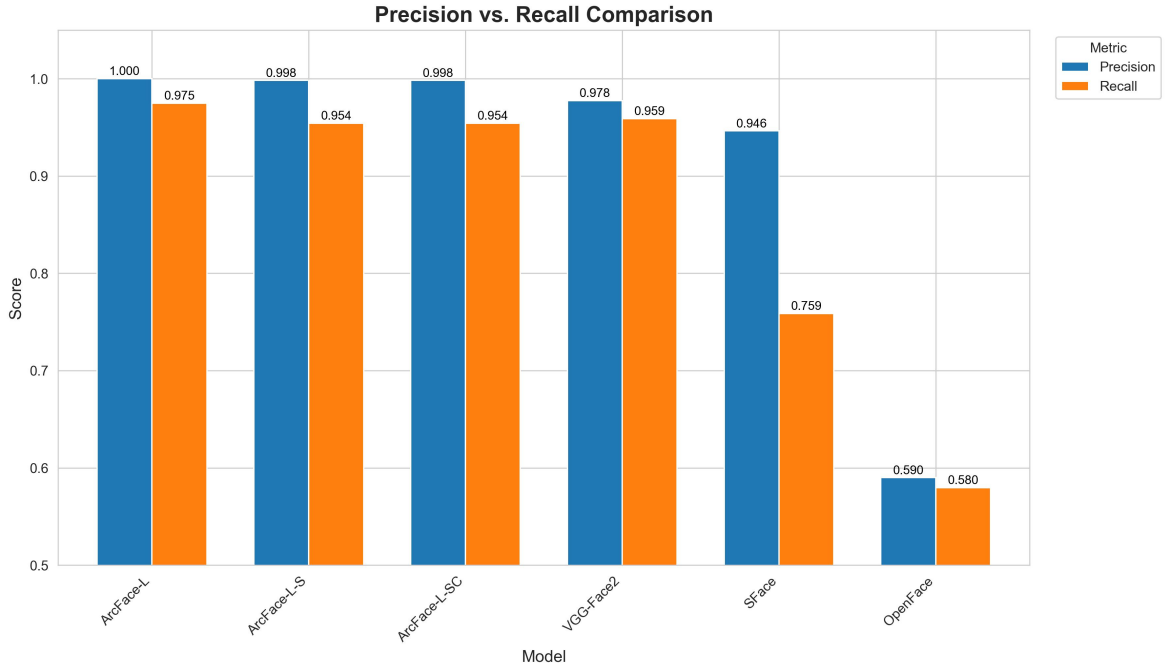


Figure 4.17: *Vergleich von Precision und Recall für verschiedene Gesichtserkennungsmodelle*

Evaluierungen wäre es demnach verlockend, zur Gesichtsdetektion Yolov8n-Face zu nutzen und die Bounding Boxes von Letzterem zum Abgleich mit der Datenbank zu verwenden, indem die Embeddings mit Buffalo-SC erstellt werden. Dies funktioniert in der Praxis jedoch weniger gut als in der Theorie. Der Grund hierfür sind vor allem die Bounding Boxes von Yolov8. Diese unterscheiden sich, auch wenn nur minimal, von denen der Buffalo-Modelle. Dies führt zu Ungereimtheiten und sowohl beim Benchmarking als auch beim Abgleich mit realen Beispielen zu deutlichen Problemen und ungenauen Ergebnissen. Des Weiteren kann ArcFaces Fähigkeit, sowohl Detektion als auch Erkennung durchzuführen, genutzt werden, um mit der gegebenen Architektur speichereffizient zu arbeiten, indem vorab die Datenbank geladen wird und dementsprechend danach Detektion und Erkennung stattfinden. Das macht das System nicht nur aus Performancesicht präziser und schneller, sondern auch aus Softwaresicht leichter lesbar. Aufgrund der genannten Gründe wird sich final für die folgende Modellauswahl entschieden:

Table 4.2: *Finale Auswahl an Modellen*

Zweck	Modell
Textgenerierung	Gemini 2.5 Flash-lite
Speech-to-Text	Faster Whisper Medium
Text-to-Speech	FastPitch
Face Detection	ArcFace Buffalo SC
Face Recognition	ArcFace Buffalo SC

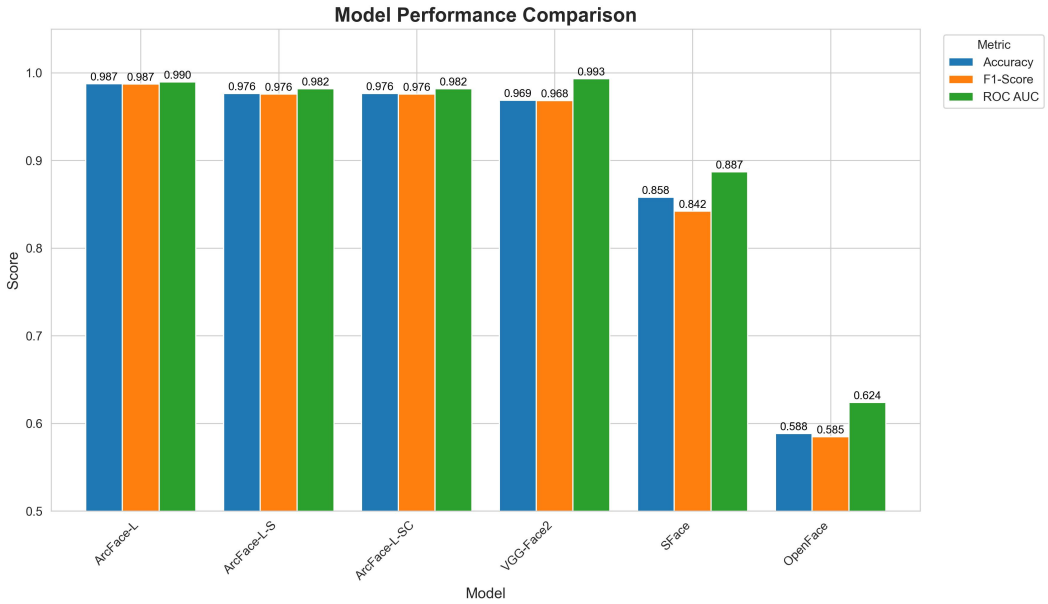


Figure 4.18: *Gesamtleistungsvergleich von Gesichtserkennungsmodellen anhand verschiedener Metriken*

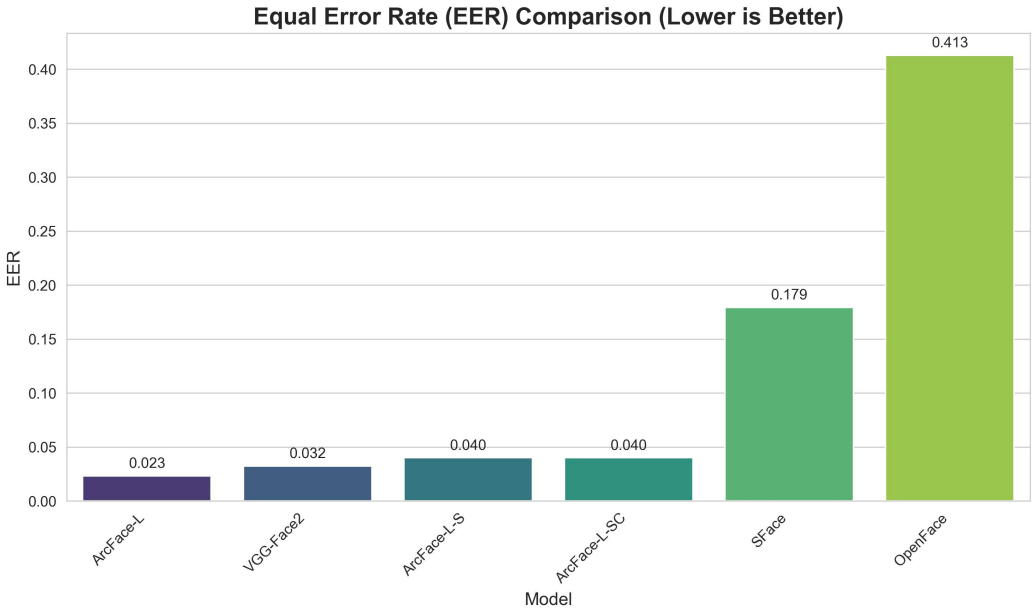


Figure 4.19: *Vergleich der Equal Error Rate (EER) für Gesichtserkennungsmodelle*

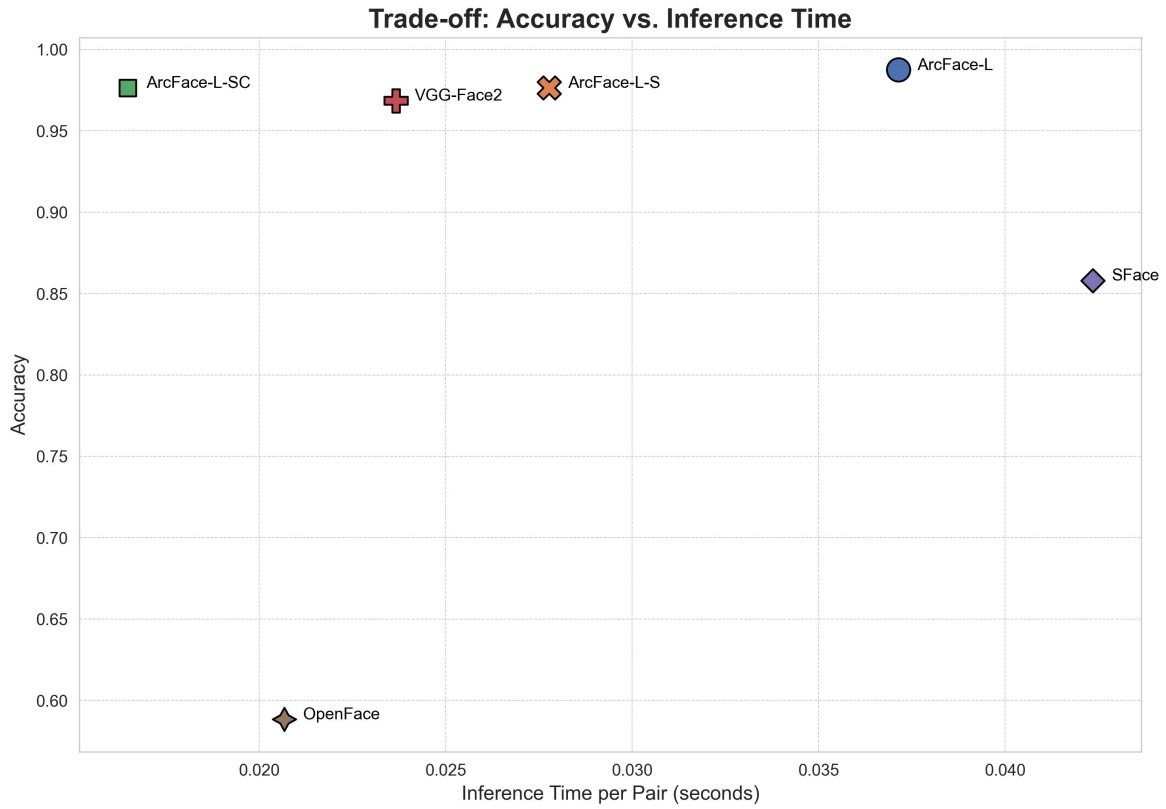


Figure 4.20: *Trade-off-Analyse für Gesichtserkennung: Genauigkeit vs. Inferenzzeit pro Bildpaar*

4.3 Evaluation und Diskussion der Ergebnisse

Mit dem Ziel, die Echtzeitfähigkeit durch Latenzminimierung in praxisrelevanten Umgebungen zu erzielen, wurden über verschiedene Modelle, Methoden und Metriken hinweg viele Ansätze verglichen. Recall und Precision für Gesichtsdetektion und -erkennung standen für die Sprachverarbeitung ERR, WER und RealTime-Factor im Fokus. Aufgrund seiner Komplexität erfordert das System ineinandergreifende, jedoch prinzipiell für sich selbst laufende Komponenten. Um diese Komponenten optimal zu wählen, kam es zu Vergleichen, die in jedem Bereich zu präzisen Entscheidungen führten. Das Ergebnis waren sorgfältige Resultate, basierend auf verschiedenen Benchmarks über zahlreiche Datensätze.
Googles Gemini LLM-Reihe verfügt über außerordentliche Stärken. Dazu zählen neben präzisen Antworten auch die Fähigkeit, schnell Texte zu generieren. Dies wurde gerade durch die Architektur und den API-basierten Cloud-Zugriff ermöglicht. Mit einer Antwortzeit von über 15 Sekunden wurde dementsprechend das 2.5 Pro Modell ausgeschlossen, und es wurde immer deutlicher, dass sich die Flash-Lite-Variante ideal eignet. Die extrem niedrige Latenz führt zu nutzerfreundlichen Konversationen. Der überaus starke Real-Time-Factor von Faster Whisper ist seinem grundlegenden Aufbau einer schnelleren Whisper-Variante zu verdanken. Die Medium-Variante trifft dabei einen idealen Sweet Spot, der trotz geringfügig besserer Werte von Groq und Vosk auch bei zunehmenden Audiolängen konstant praktikable Ergeb-

nisse liefert. Auf der anderen Seite befanden sich die Methoden zur Sprachsynthese. Dabei stachen FastPitch und Pyttsx3 deutlich hervor und erwiesen sich als klare Gewinner. Mit einer enorm starken Generationsgeschwindigkeit und authentischen Stimmen erwies sich FastPitch als optimales Element. Diese Wahl sorgt dafür, dass der RTF in der Praxis und unter realen Bedingungen und Konversationen stets unter 0,1 bleibt und eine Bearbeitungszeit von unter einer Sekunde erzielt wird.
Die in Abbildung 4.15 gezeigten Ergebnisse führten zur Annahme, dass MediaPipe aufgrund seiner Schnelligkeit ein Kandidat sein könnte. Dies wurde jedoch durch die Betrachtung des mAP entkräftet. Die hohe Performance parallel zur schnellen Exekution verzeichnete das Yolov8n-Face- und ArcFace-Buffalo-SC Modell unter den Top-Kandidaten. Doch der wichtigere Blick lag, mit Fokus auf Face Recognition, beim Trade-off zwischen Accuracy und Inferenzzeit, wie in Abbildung 4.20 dargestellt. Mit seiner schnellen Ausführung zeigte sich die Effizienz des abgespeckten Modells von Buffalo, ohne stark an Genauigkeit zu verlieren. Andere Modelle wie SFace sind im Vergleich zwar schneller, zeigen jedoch deutliche Nachteile in Theorie und Praxis. Aufgrund der Vereinfachung und der in sich vorliegenden Modelle wurde entschieden, das ArcFace SC Modell sowohl zur Detektion als auch zur Erkennung zu verwenden. Das lag unter anderem auch daran, dass die Bounding Boxes von Yolo nicht stimmig mit den Embeddings der ArcFace-Methodik waren, sodass kaum zuverlässige Ergebnisse erzielt werden konnten. Damit werden in Kombination mit der speichereffizienten Implementierung stets über 100 Frames pro Sekunde im Browser erzielt. Anhand eines Beispiel-Interviews der Obamas [198] lässt sich das Ergebnis trotz niedriger Videoqualität einwandfrei beobachten. Das einzige in der Datenbank hinterlegte Bild von Obama reicht bereits aus, um bekannte Gesichter, in diesem Fall Barack, zu erkennen und unbekannte Gesichter, hier Michelle Obama, in Echtzeit zu retuschieren und unkenntlich zu machen.



Figure 4.21: *Anwendung der Face Detection und Recognition*

Die finale Modellauswahl basiert auf einer gezielten Optimierung für Latenz und Praxisrelevanz. Für die Textgenerierung wurde Gemini 2.5 Flash-lite aufgrund seiner minimalen Antwortzeit gewählt, die für eine flüssige Konversation unerlässlich ist. Faster Whisper Medium bietet als Speech-to-Text-Modell den besten Kompromiss aus Genauigkeit und konstanter Geschwindigkeit, während FastPitch durch schnelle und natürliche Sprachsynthese überzeugt. Im anspruchsvollen Bereich der Gesichtsverarbeitung wird ArcFace Buffalo SC für Detektion und Erkennung eingesetzt. Obwohl alternative Detektoren eine höhere mAP aufweisen, erwies sich die Synergie eines integrierten Modells in der Praxis als robuster, da hierdurch Inkonsistenzen vermieden werden. Diese für den Praxiseinsatz optimierte Modellauswahl bildet die Grundlage für die nun folgende Analyse des Gesamtsystems in Kapitel 5.

Kapitel 5

Diskussion

5.1 Interpretation und Einordnung der Arbeit

5.1.1 Ergebnisse

Als Grundlage einer effizienten Systemarchitektur erwies sich Python in Kombination mit Selenium als ideale Wahl. Die Möglichkeit, menschliches Verhalten zu simulieren, ist eine grundlegende Komponente, die die Arbeit aller darauffolgenden Komponenten ermöglicht.
Die ausführliche Evaluation in Kapitel 4 hat gezeigt, dass die zentralen Anforderungen an ein latenzarmes System durch die Wahl geeigneter Komponenten erfüllt werden. Mit einer direkten API-Schnittstelle wurden im Schnitt Antworten in 1,1 Sekunden erzielt, trotz der Tatsache, dass es sich um ein vollständiges Large Language Model handelt. Dies ist vor allem auf die herabgestufte Version der Parameter zurückzuführen. Ein Vergleich mit der 2.5 Pro Variante zeigt nämlich eine erhöhte Geschwindigkeit von bis zu 1400 Prozent.
Die Spracherkennung und Transkription ist vor allem in Bezug auf die Textgenerierung und -synthese wichtig. Eine Voraussetzung neben der Geschwindigkeit war ein niedriger RealTime-Faktor als Maß der Verarbeitungsgeschwindigkeit. Zusammen mit der Transkriptionsdauer erwies sich das Faster Whisper-Modell als überdurchschnittlich gut. Trotz schlechterer Werte bei den Character- und Word-Error-Rates, die um bis zu 69 Prozent gegenüber Vosk bei der Nutzung von Medium schlechter waren, waren die rechenoptimierten Versionen von Tiny bis Large nicht nur im Hinblick auf die Ausführung, sondern auch auf die Skalierbarkeit deutlich im Vorteil. Ein Test mit realen Umgebungsvariablen zeigte trotz der vorliegenden Werte zuverlässige Ergebnisse. Das Medium-Modell erwies sich als idealer Kompromiss zwischen Genauigkeit und Geschwindigkeit und war im Hinblick auf den RTF nahezu 76 Prozent besser.
Ein letzter Schritt in der Sprachverarbeitung beschäftigt sich mit der authentischen und effizienten Generierung von Sprache. Im Gegensatz zu den vorherigen Vergleichen hat sich FastPitch direkt über alle Metriken als optimale Wahl erwiesen. Mit der schnellsten Generierungszeit und der zugleich menschenähnlichsten Stimme konnte klar aufgezeigt werden, warum dies als Element im System vorhanden sein muss.

5.1.2 Rückblick auf die Forschungsfragen

Zu Anfang wurden drei zentrale Forschungsfragen aufgestellt, die es zu untersuchen galt. Aufgrund des geschlossenen Entwickler Ökosystems der RayBan Meta Glasses war eine direkte

Anbindung nicht möglich. Als Lösung musste eine eigene Architektur entworfen werden, die sich über Facebook Messenger an den verfügbaren Daten bedient hat. Zentraler Bestandteil war dafür Pythons Selenium Bibliothek, die den gesamten Prozess ermöglichte. Die Zuverlässigkeit dieser prototypischen Pipeline konnte auf die Producer-Consumer-Architektur zurückgeführt werden, die eine hohe Performance aufgrund der Entkoppelung von Erfassung und Analyse ermöglicht hat.
Das Ermöglichen einer latenzarmen Sprachverarbeitung wurde durch mehrere Faktoren erreicht. Zum einen kamen die cloudbasierten Schnittstellen von Gemini zum Einsatz, die eine große Menge an Rechenlast abgenommen haben. Allein dieser Punkt führte zu deutlichen Steigerungen in der Textverarbeitung. Neben der reinen Textgenerierung spielen jedoch auch Komponenten zur automatischen Spracherkennung und -synthese eine Rolle. Mit einer bewussten, auf Trade-offs basierenden Entscheidung für das Fast Whispers Medium Model ermöglichten extrem niedrige und zugleich konstante Latenzen, wie in Abbildung 4.9 und 4.8 zu sehen ist. Neben dem Echtzeit-Erlebnis waren auch authentische Stimmen von Bedeutung, um mit bekannten Systemen wie Siri und Alexa zu ähneln. FastPitch konnte nicht nur aufgrund seiner menschlich ähnlichen Stimme überzeugen, sondern erwies sich auch bei bekannten Sprachsynthese Metriken wie der Geschwindigkeit als deutlicher Sieger.
Der für sich entscheidende Part war die Bewahrung der Privatsphäre bei der Verarbeitung von Bildern aus Anrufen. Neben einer herkömmlichen Detektion von Gesichtern war es ebenso von Wichtigkeit, diese mit einer vordefinierten Datenbank abzugleichen und gegebenenfalls auszublenden. Die Anonymisierung spielt dabei eine besondere Rolle. Diese muss jederzeit Nach Wunsch erfüllt werden, um dem Privacy-by-Design Ansatz gerecht zu werden. Aufgeteilt in zwei Parts wurden neben der Detektion auch unterschiedliche Erkennungsmodelle getestet. Aufgrund der privatsphärenbestimmenden Eigenschaften waren hier neben der Anzahl der verarbeiteten Frames pro Sekunde der Recall wichtig, um eher zum Ausblenden zu tendieren als Gesichter anzuzeigen. Die vortrainierten Modelle Buffalo von ArcFace, insbesondere das kleine SC-Modell, erwiesen sich neben der Geschwindigkeit auch bei der praktischen Analyse als zuverlässige Wahl.
Somit konnten alle drei ursprünglichen Fragen nicht nur im Detail untersucht, sondern auch tiefgründig beantwortet und gelöst werden.

5.2 Kritische Reflexion und Herausforderungen

Mit einem kritischen Blick auf die Infrastruktur, die verwendeten Komponenten und die Methoden lassen sich einige Schwierigkeiten entdecken, die nicht zu unterschätzen sind. Dies betrifft neben den Methoden auch prinzipielle Elemente des Systems, die sowohl anfällig als auch nicht von langer Lebensdauer sind.
Angefangen mit der Webautomatisierung lassen sich einige Schwierigkeiten und Probleme finden. Zum einen besteht eine starke Abhängigkeit von Selenium. Durchgehende Updates an der Bibliothek sorgen für neue Methoden, Prinzipien und gezwungene Vorgehensweisen. Selenium ist auch bekannt für seine Ressourcenintensive Auslastung und Nutzung, die für ein latenzarmes System eigentlich fundamental ist. Abgesehen davon gibt es ebenfalls Gründe, die außerhalb der technischen Macht des Erstellers und Nutzers liegen. Dazu gehören die ständigen Updates von Facebook bezüglich der Messenger-Oberfläche. Identifizierte Elemente ändern ständig Klassennamen oder auch ID-Tags, was dazu führt, dass diese nicht

mehr gefunden werden können und damit eine Interaktion nicht ermöglicht wird. neben den Frontend-Änderungen gibt es auch Sicherheitsmaßnahmen, deren Präsenz nicht ohne Grund ist. Um böswillige Bots, darunter eventuell das hier genutzte Programm, zu vermeiden, werden bestimmte Mechanismen eingesetzt. Für diesen Fall hat es gereicht, einen accountspezifischen Pin per Tastensimulation einzugeben. Bei der Implementierung von Captchas oder 2-Faktor-Authentifizierungen wird es jedoch nahezu unmöglich, diese vollkommen automatisiert zu umgehen. Eine Möglichkeit besteht zwar in der Nutzung von Browserprofilen, die sowohl Cookies als auch andere Metadaten speichern und somit nicht jedes mal auf neue einen login erwarten, doch sind diese technisch weitaus aufwendiger. Darüber hinaus ist der künstliche Browser anfällig gegenüber Fehlern und stürzt bei Ungereimtheiten komplett ab. Dies zwingt den Nutzer, das Programm erneut von Anfang an zu starten, was neben Geduld auch eine Suche nach dem ursprünglichen Fehler erfordert. Doch aufgrund der fehlenden API-Schnittstelle gegenüber privaten Accounts ist es anders nicht möglich. Daher ist unter der Voraussetzung der RayBan Meta Glasses dieser erwähnte Prototyp die einzige Möglichkeit.
Eine Philosophie dieser Architektur besteht darin, verschiedene Komponenten zusammenzuführen, die in ihren jeweiligen Bereichen optimale Ergebnisse erzielen. Doch spätestens bei der Beobachtung verschiedener Face Detection Modelle ist aufgefallen, dass die Kombination nicht willkürlich stattfinden kann. Trotz der Leistung von Yolov8n-Face war direkt ersichtlich, dass aufgrund abweichender Bounding Boxes die Gesichtserkennung im darauffolgenden Schritt unpraktikabel war. Als Ausweichschritt musste dementsprechend sowohl bei Detection als auch bei Recognition auf dasselbe Modell von ArcFace zurückgegriffen werden. Ídealerweise handelte es sich bei Letzterem aus Performancesicht um eine geeignete Wahl, soll jedoch aufzeigen, welche kritischen Punkte bei der Kombination willkürlicher Komponenten entstehen.
Aufgrund der Nutzung herkömmlicher Geräte musste sichergestellt werden, dass die Verarbeitung so schnell und präzise wie möglich abläuft. mittels Nvidia's CUDA und einer starken CPU konnten Bildaufnahme und Analyse parallelisiert werden. Doch auch hier ergaben sich bereits schnell Grenzen. In Bildern mit einer Handvoll Gesichtern erlangte das System über 100 frames pro Sekunde in der Analyse. Dies änderte sich rasant, wenn signifikant mehr Gesichter vertreten waren. Das führte unter anderem dazu, dass sogar in mittlerweile einstelligen Bereichen Frames aufzufinden waren. Daneben ist auch die Qualität des Bildmaterials von Bedeutung. Aufgrund der Aufnahmen aus Videotelefonaten unter Wlan war dies in Ordnung, zeigte jedoch schnell Anfälligkeiten gegenüber schlechterer Internetverbindung auf.
Die Arbeit mit Modellen aus verschiedenen Richtungen zeigte, dass unterschiedliche Datensätze verwendet wurden. Trotz Vielfalt und Variabilität hinsichtlich unterschiedlicher Attribute stoßen alle Modelle an ihre Grenzen. Dies kann auf einen Bias in den Trainingsdaten zurückgeführt werden. Dabei reicht es teilweise schon aus, in unbekannten Sprachen oder sogar Dialekten zu sprechen, die in der Praxis zu wesentlich schlechteren Ergebnissen führen. Auch dies muss beachtet werden.
Fragen in Richtung Sicherheit kommen gerade bei der Betrachtung der Personenbezogenen Daten auf. Mit einem Fokus auf Privatsphäre stellt sich die Frage, wer anfänglich die Referenzbilder von Personen kontrolliert und die Zustimmung überwacht.

5.3 Einordnung in den Stand der Technik

Nach der Interpretation der eigenen Ergebnisse ordnet dieser Abschnitt die vorliegende Arbeit in den Kontext der in Kapitel 2 dargestellten verwandten Forschungen ein. Ziel ist es, den eigenen Beitrag im Vergleich zu bestehenden Ansätzen herauszuarbeiten und die wesentlichen Unterschiede sowie Gemeinsamkeiten aufzuzeigen.

Fokus auf Systemintegration statt Algorithmenoptimierung

Ein wesentliches Unterscheidungsmerkmal dieser Arbeit im Vergleich zu vielen in Kapitel 2.6 vorgestellten Forschungsprojekten, wie beispielsweise Vinci oder EgoLife, liegt im Fokus auf der Systemintegration anstelle der reinen Algorithmenoptimierung. Während in der akademischen Forschung häufig die Entwicklung neuartiger Modelle oder die Verbesserung spezifischer Metriken (z.B. Erkennungsgenauigkeit) im Vordergrund steht, liegt der Beitrag dieser Arbeit in der praxisorientierten Auswahl, Kombination und Implementierung von State-of-the-Art-Komponenten zu einem funktionalen Gesamtsystem. Der Schwerpunkt ist somit weniger die algorithmische Neuerfindung als vielmehr der Nachweis der Machbarkeit einer komplexen Echtzeit-Pipeline auf gängiger Consumer-Hardware.

Priorisierung von Latenz für die Praxistauglichkeit

Die in Kapitel 4 getroffene Modellauswahl bestätigt und unterstreicht die wachsende Bedeutung der Latenz als kritischen Faktor für die Praxistauglichkeit von interaktiven KI-Systemen. Die Entscheidung, das Faster Whisper Medium-Modell trotz einer leicht höheren Wortfehlerrate im Vergleich zu Cloud-basierten Lösungen oder dem VOSK-Modell zu verwenden, ist ein direkter Trade-off zugunsten eines nahtlosen Echtzeit-Erlebnisses. Diese Priorisierung der Verarbeitungsgeschwindigkeit (RTF kleiner 0,1) über die maximale Genauigkeit steht im Einklang mit Forschungsansätzen, die die User Experience in den Mittelpunkt stellen, und grenzt sich von rein metrik-getriebenen Benchmarks ab.

Pragmatische Lösung für geschlossene Ökosysteme

Des Weiteren adressiert die Arbeit eine praktische Herausforderung, die in der Literatur zu Wearable-Technologien oft nur am Rande behandelt wird: den Umgang mit geschlossenen technologischen Ökosystemen. Die Nutzung von Browser-Automatisierung via Selenium zur Ansteuerung der Meta Ray-Ban-Glasses ist eine pragmatische Lösung, die demonstriert, wie Prototyping und Forschung auch ohne offizielle Entwicklerschnittstellen oder APIs möglich sind. Dies stellt einen relevanten Lösungsansatz für die anwendungsorientierte Forschung dar, die sich häufig mit proprietärer Hard- und Software konfrontiert sieht.

Bestätigung und Erweiterung bestehender Ansätze

Die Arbeit erfindet die Gesichtsverarbeitung nicht neu, sondern bestätigt die Leistungsfähigkeit moderner Architekturen wie ArcFace in einem anspruchsvollen Echtzeit-Anwendungsszenario. Darüber hinaus leistet sie einen wichtigen praktischen Beitrag, indem sie aufzeigt, dass die theoretisch optimale Kombination von Einzelkomponenten – dem performantesten Detektor (YOLOv8n-Face) und dem präzisesten Erkenner (ArcFace) – in der Praxis aufgrund von

Inkompatibilitäten bei den Datenstrukturen (Bounding Boxes vs. Embeddings) nicht zwangsläufig zur besten Gesamtlösung führt. Diese Erkenntnis, dass ein integriertes System wie ArcFace Buffalo SC in der Praxis robuster sein kann, erweitert den Stand der Technik um eine wichtige, implementierungsnahe Dimension. Zusammenfassend lässt sich sagen, dass sich diese Masterarbeit nicht durch die Entwicklung einer isolierten, neuen Methode in den Stand der Technik einordnet, sondern durch ihren holistischen und anwendungsorientierten Ansatz. Sie schlägt eine Brücke zwischen theoretischen SOTA-Modellen und den realen Herausforderungen einer Implementierung – Latenz, Systemintegration, geschlossenen Plattformen und praktischen Inkompatibilitäten. Damit liefert sie einen wertvollen Beitrag für die Realisierung zukünftiger, praxisnaher Assistenzsysteme auf Smartglasses.

5.4 Limitierungen der Arbeit

Im Rahmen einer transparenten wissenschaftlichen Auseinandersetzung ist es wesentlich, die Grenzen der vorliegenden Arbeit zu definieren. Trotz der erfolgreichen Implementierung eines funktionsfähigen Prototyps unterliegen das Vorgehen und die Ergebnisse bestimmten Einschränkungen. Diese zu benennen, dient nicht nur der korrekten Einordnung der erzielten Resultate, sondern zeigt auch potenzielle Ansatzpunkte für zukünftige Arbeiten auf.

Abhängigkeit von der Browser-Automatisierung

Eine zentrale technische Limitierung ist die Nutzung von Selenium zur Steuerung der Messenger-Anwendung. Dieser Ansatz ist von der grafischen Benutzeroberfläche von Facebook Messenger abhängig. Zukünftige Änderungen am Design oder an den internen Element-IDs der Webseite könnten die Funktionsfähigkeit der entwickelten Pipeline beeinträchtigen oder gar unterbrechen. Die Lösung ist daher in ihrer langfristigen Stabilität fragil und nicht für einen produktiven Einsatz geeignet, solange keine offizielle API zur Verfügung steht.

Spezifische Hard- und Softwarekopplung

Das entwickelte System ist eng an das Ökosystem der Meta Ray-Ban Smartglasses und die Facebook Messenger-Plattform gekoppelt. Die Implementierung und die Evaluationsergebnisse sind somit nicht ohne Weiteres auf andere Smartglasses oder Kommunikationsdienste übertragbar. Die Generalisierbarkeit der konkreten Lösung ist dadurch eingeschränkt, auch wenn die grundlegenden Konzepte der Systemarchitektur und der Modellauswahl von allgemeiner Relevanz sind.

Umfang der Evaluation

Die Evaluation der Systemkomponenten wurde unter kontrollierten Bedingungen mit standardisierten, öffentlichen Datensätzen (z. B. WIDER FACE, LibriSpeech) durchgeführt. Diese Vorgehensweise gewährleistet zwar die Reproduzierbarkeit und Vergleichbarkeit der Ergebnisse, bildet jedoch nur begrenzt die Komplexität realer Anwendungsszenarien ab. Faktoren wie stark variierende Lichtverhältnisse, laute Umgebungsgeräusche, unterschiedliche Dialekte oder unvorhergesehene Kopf- und Kamerabewegungen wurden nicht systematisch getestet.

Die in Kapitel 4 dokumentierte Performance stellt daher eine Baseline dar, die in der Praxis abweichen kann.

Prototypischer Charakter des Systems

Die vorliegende Implementierung hat den Charakter eines "Proof-of-Concept". Der Fokus lag auf der Realisierung der Kernfunktionalitäten und der Optimierung der Echtzeit-Performance. Aspekte wie eine robuste Fehlerbehandlung, eine benutzerfreundliche Konfigurationsoberfläche oder die Skalierbarkeit der Gesichtserkennungs-Datenbank für eine große Anzahl von Personen wurden bewusst vernachlässigt. Das System ist in seiner jetzigen Form für den Einsatz durch einen einzelnen Anwender in einer Testumgebung ausgelegt.

Kapitel 6

Fazit und Ausblick

6.1 Zusammenfassung der Arbeit

Mit den limitiert zur Verfügung stehenden Möglichkeiten von Smart Glasses im Allgemeinen sollte am Beispiel betrachtet werden, wie diese durch externe Modulitäten so erweitert werden können, dass ein smartes Assistenzsystem mit intuitiver, gestenloser Bedienung zur Verfügung steht. Das Ziel war also die Implementierung einer latenzarmen, intelligenten Assistenz zur Sprachinteraktion, die unter Wahrung der privatsphäre Bilder verarbeiten und analysieren konnte. Dies sollte anhand der handelsüblichen Meta Glasses von Ray Ban realisiert werden, da diese durch ihre eingebaute Kamera mit Mikrofon und Lautsprecher die idealen Voraussetzungen mit sich brachte.
Für die Realisierung wurde mittels Python und Selenium eine Pipeline gebaut, die die automatisierte Interaktion ermöglicht hat. Die Sprachverarbeitungspipeline ernährt sich von Transkriptionen, Textgenerierung und Sprachsynthese. Diese drei Bereiche wurden detailliert auf verschiedene Ansätze und Methoden im Hinblick auf die verfügbaren Modelle getestet. Für die Bildverarbeitung war eine Betrachtung aus zwei Perspektiven wichtig. Dazu zählen neben der Gesichtserkennung auch die Gesichteridentifizierung als fundamentaler Bestandteil zum Schutz der Privatsphäre.
Für die Textgenerierung erwies sich Geminis 2.5 Flash Lite als zentrale Komponente zur Generierung von Text und zur Ermöglichung eines flüssigen Chatverlaufs. Mit einer Antwortzeit von durchschnittlich 1,1 Sekunden war es das schnellste Modell der neuen Reihe. Darüber hinaus waren bemerkenswerte RTF Werte bei der Transkription´und Sprachsynthese zu sehen. Mit niedrigen Latenzen konnten das Medium Modell von Faster Whispers in Kombination mit der Fähigkeit von FastPitch, authentisch generierte Stimmen zu erzeugen, ideale Bedingungen schaffen. Eine reibungslose Gesichtserkennung und Identifikation mit über 100 Frames pro Sekunde wurde vor allem durch die ArcFace-Buffalo-Modellreihe geschaffen. Mit der SC variante konnten Erwartungen erreicht werden. Dabei ist wichtig zu erwähnen, dass die Synergie zwischen den Modellen weitaus wichtiger war als die isolierte Spitzenleistung, wie anhand des Beispiels von Yolov8n-face zu sehen war.

6.2 Beantwortung der Forschungsfragen

Im Rahmen dieser Arbeit wurden drei zentrale Forschungsfragen formuliert, die durch die Konzeption, Implementierung und Evaluation des prototypischen Systems beantwortet werden sollten. Die gewonnenen Erkenntnisse werden im Folgenden systematisch dargelegt. Die erste Forschungsfrage befasste sich mit der Möglichkeit, eine zuverlässige Pipeline zur kontinuierlichen Sammlung und Verarbeitung von Daten zu etablieren, obwohl die verwendete Hardware der Meta Ray-Ban Glasses keine offizielle Entwicklerschnittstelle bereitstellt. Die im Rahmen dieser Arbeit entwickelte Architektur liefert den Nachweis, dass eine solche Pipeline erfolgreich realisiert werden kann. Durch den pragmatischen Einsatz von Browser-Automatisierung mittels Selenium wurde eine stabile Brücke zum geschlossenen Ökosystem des Geräts geschlagen. Die entscheidende Komponente für die Zuverlässigkeit und Performance war jedoch die Implementierung einer Producer-Consumer-Architektur, welche die Datenerfassung von der rechenintensiven Analyse entkoppelt. Dieser Aufbau verhinderte effektiv Engpässe und gewährleistete einen robusten und kontinuierlichen Datenfluss. Die zweite Forschungsfrage untersuchte, inwieweit Speech-to-Text- und Text-to-Speech-Modelle das Interaktionserlebnis in Bezug auf Latenz steuern. Die durchgeführte Evaluation hat eindeutig gezeigt, dass die Latenz der entscheidende Faktor für eine natürliche und als flüssig empfundene Mensch-Maschine-Kommunikation ist. Die bewusste Kombination aus einem API-basierten, aber hochoptimierten Large Language Model für die Textgenerierung und lokal ausgeführten, effizienten Modellen für die Spracherkennung und Sprachsynthese ermöglichte eine Gesamtverarbeitungskette mit minimalen Verzögerungen. Mit Reaktionszeiten, die konsistent unter einer Sekunde für die Sprachverarbeitung lagen, und einem Real-Time-Factor von unter 0.1, wurde ein Echtzeit-Erlebnis geschaffen, das für die praktische Akzeptanz eines solchen Assistenzsystems als fundamental erachtet wird. Die dritte und letzte Forschungsfrage zielte darauf ab, wie eine Gesichtserkennung unter Wahrung der Privatsphäre realisiert werden kann. Der entwickelte Prototyp demonstriert, dass dies durch eine durchdachte Kombination aus architektonischen Entscheidungen und der gezielten Auswahl von Algorithmen möglich ist. Die Wahrung der Privatsphäre wurde durch drei wesentliche Maßnahmen sichergestellt: Erstens erfolgt die gesamte Bildverarbeitung ausschließlich lokal auf dem Endgerät des Anwenders, sodass sensible Bilddaten zu keinem Zeitpunkt an externe Server gesendet werden. Zweitens wurde eine Funktion zur Anonymisierung implementiert, die alle Gesichter, die nicht explizit in einer lokalen Datenbank hinterlegt sind, automatisch unkenntlich macht. Drittens erwies sich die Wahl des ArcFace Buffalo SC Modells als ideal, da es neben einer hohen Verarbeitungsgeschwindigkeit auch eine für den Datenschutz kritische hohe Recall-Rate aufweist. Dadurch wird das Risiko minimiert, ein zu schützendes Gesicht fälschlicherweise nicht zu erkennen und somit die Integrität der datenschutzwahrenden Funktion gestärkt.

6.3 Ausblick auf zukünftige Arbeiten

Obwohl der im Rahmen dieser Masterarbeit entwickelte Prototyp die grundsätzliche Machbarkeit des Konzepts erfolgreich demonstriert, legt die gewählte Architektur auch die wichtigsten Ansatzpunkte für zukünftige Arbeiten offen. Diese lassen sich in die Bereiche der technischen Fundamentierung, der funktionalen Erweiterung und der ethischen Vertiefung gliedern. Eine entscheidende Weiterentwicklung wäre die Überwindung der technischen Limitierungen

des aktuellen Systems. Die zentrale Schwachstelle stellt die Anbindung der Hardware über Browser-Automatisierung dar, die naturgemäß fragil gegenüber Änderungen an der Benutzeroberfläche der Messenger-Anwendung ist. Zukünftige Projekte sollten daher die Migration des Systems auf eine Hardware-Plattform evaluieren, die über ein offenes Software Development Kit oder eine dokumentierte API verfügt, um eine robustere und performantere Datenanbindung zu gewährleisten. Darüber hinaus könnte der Prototyp zu einer anwendungsreifen Lösung ausgebaut werden. Dies umfasst die Implementierung einer robusten Fehlerbehandlung, die Schaffung einer benutzerfreundlichen Konfigurationsoberfläche zur Verwaltung der Gesichtsdatenbank sowie die weitere Optimierung der Modelle durch Techniken wie Quantisierung, um den Betrieb auf noch leistungsschwächeren, mobilen Endgeräten zu ermöglichen. Aufbauend auf einer stabilisierten technischen Grundlage eröffnen sich weitreichende Möglichkeiten zur funktionalen Erweiterung des Assistenten. Das größte Potenzial liegt in der Realisierung einer echten multimodalen Fusion, bei der die bisher noch weitgehend isoliert arbeitenden Pipelines für Sprach- und Bildverarbeitung zu einem integrierten System verschmelzen. Ein solches System könnte kontextbezogene Fragen wie "Wen sehe ich gerade?"direkt beantworten oder auf Objekte im Sichtfeld reagieren. Ein weiterer vielversprechender Entwicklungsschritt liegt in der Transformation des reaktiven Assistenten zu einem proaktiven Begleiter. Durch die kontinuierliche Analyse des egozentrischen Videostreams könnte das System antizipieren, welche Informationen für den Nutzer relevant sind, und diese proaktiv bereitstellen – beispielsweise an den Namen einer Person erinnern, die in das Sichtfeld tritt. Neben den technischen Erweiterungen werfen solche fortgeschrittenen Assistenzsysteme auch wichtige ethische und datenschutzrechtliche Fragestellungen auf, die Gegenstand zukünftiger Forschung sein müssen. Ein zentraler Aspekt ist hierbei das Einwilligungsmanagement. Es bedarf der Entwicklung transparenter und benutzerfreundlicher Mechanismen, mit denen Personen ihre Zustimmung zur Speicherung und Verarbeitung ihrer biometrischen Daten erteilen, verwalten und jederzeit widerrufen können. Ferner muss das Problem potenzieller demografischer Verzerrungen in den Erkennungsmodellen adressiert werden. Systematische Analysen zur Fairness der Modelle sowie die Erforschung von Techniken zur Minderung von Bias sind unerlässlich, um sicherzustellen, dass die Technologie für alle Nutzergruppen gleichermaßen zuverlässig und gerecht funktioniert.

List of Acronyms

AI Artificial Intelligence

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

AR Augmented Reality

ASR Automatic Speech Recognition

AUC Area Under Curve

BERT Bidirectional Encoder Representations from Transformers

CER Character Error Rate

CNN Convolutional Neural Network

CPS Characters Per Second

CPU Central Processing Unit

CUDA Compute Unified Device Architecture

cuDNN CUDA Deep Neural Network library

DDCS Double Decoder Consistency

DL Deep Learning

DOM Document Object Model

DSP Digital Signal Processing

EER Equal Error Rate

Ego4D Egocentric 4D Perception

FAR False Acceptance Rate

FN False Negative

FP False Positive

FPR False Positive Rate

FPS Frames Per Second

FRR False Rejection Rate

GAN Generative Adversarial Network

GPS Global Positioning System

GPU Graphics Processing Unit

GPT Generative Pre-trained Transformer

HOG Histogram of Oriented Gradients

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IMU Inertial Measurement Unit

IoU Intersection over Union

JSON JavaScript Object Notation

KI Künstliche Intelligenz

LBP Local Binary Patterns

LFW Labeled Faces in the Wild

LLM Large Language Model

mAP mean Average Precision

ML Machine Learning

MoE Mixture of Experts

NLP Natural Language Processing

PaLM Pathways Language Model

PIN Personal Identification Number

RAM Random Access Memory

RLHF Reinforcement Learning from Human Feedback

ROC Receiver Operating Characteristic

RTF Real-Time Factor

RVQ Residual Vector Quantization

SDK Software Development Kit

SOTA State-of-the-Art

SSD Single Shot MultiBox Detector

STT Speech-to-Text

SVM Support Vector Machine

TDNN Time Delay Neural Network

TDNN-F Factorized Time Delay Neural Network

TN True Negative

TP True Positive

TPR True Positive Rate

TPU Tensor Processing Unit

TTS Text-to-Speech

VGG Visual Geometry Group

VR Virtual Reality

VRAM Video Random Access Memory

W3C World Wide Web Consortium

WER Word Error Rate

WFST Weighted Finite-State Transducer

WPM Words Per Minute

WPS Words Per Second

YOLO You Only Look Once

Abbildungsverzeichnis

1.1 Meta Quest 3 und die Ray-ban Meta Glasses [5] mit Quests Displaysicht [6] . 2

2.1 Technische Komponenten der Meta Glasses [13] 5
2.2 Prinzipieller Aufbau einer Transformer-Architektur in Large Language Modellen [43] 8
2.3 Inklusion von Experten [55] 9
2.4 Der Vergleich herkömmlicher und Expert integrierter Transformer [55] 10
2.5 Prinzip und Aufbau der VOSK-Architektur 12
2.6 Architektur des Basis Whisper Modells [72] 13
2.7 Direkter Vergleich zwischen Basis und Turbo Variante des Whisper Modells mit Fokus auf die Anzahl der Decoder-Schichten [79] 14
2.8 Architektur und Komponenten von Tacotron 2 DDCS [100] 15
2.9 Architektur und Komponenten von Sesame CSM 1b [105] 16
2.10 Architektur und Komponenten von FastPitch [107] 17
2.11 Prinzipieller Algorithmus bei Haar-Kaskaden [136] und Dlib Hog [134] 20
2.12 Vielzahl an Schichten einer Deep Neural Network Single Shot Detection Architektur [142] 21
2.13 Besonderer Fokus der additien Angular Margin Loss-Funktion im Arcface Aufbau [155] 22
2.14 Dreifacher Embedding Ansatz bei OpenFace [157] 23
2.15 Aufbau und Architektur von VGG2 [159] 24
2.16 Effizienzoptimierter Aufbau von SFace [159] 25
2.17 Ausschnitte aus dem Ego4D Datensatz mit Fokus auf Aufnahmen aus der Ego-Perspektive [167] 27

3.1 Aufbau des Systems mit Unterteilung in externe und interne Interaktion . . . 32
3.2 Methoden der Browser Klasse zum initialen Start 35
3.3 Initialer Screen zur Eingabe nach Start des Browsers 37
3.4 Methoden der Browser Klasse für Interaktion und Verarbeitung 37
3.5 State nach erfolgreichem Login 38
3.6 Interaktive Pop-Up Elemente und Fenster bei eingehendem Anruf 38
3.7 Alle notwendigen Elemente zur Verarbeitung von Sprache 41
3.8 Verarbeitungsschleife der Sprache 42
3.9 Hauptschleife der Audioaufnahme mit Fokus auf aktive Stimmerkennung durch Lautstärkeschwellen 43

3.10 Initialer Start der Bereitstellung des Browserfenster-Inhalts 44
3.11 Prozesswahl der Klasse in Abhängigkeit des gewählten Modus 45
3.12 Zusammenführung des Entscheidungbaums zur finalen Analyse per Bild und Ton 46
3.13 Zusammenspiel der Komponenten zur Optimierung der Rechenleistung 47
3.14 Methoden Gesichtsverarbeitung für Start und Ende im laufenden Programm . 48

4.1 Beispiele aus dem Wider Face Datensatz mit einer Vielfalt an Bildern[191] . . 51
4.2 Typische Bilder beim Labeled Faces in the Wild Datensatz [193] 52
4.3 Durchschnittliche Antwortlänge verschiedener Gemini-Modelle in Zeichen . . . 58
4.4 Antwortlänge der Gemini-Modelle pro Testfrage 59
4.5 Durchschnittliche Antwortzeit verschiedener Gemini-Modelle in Sekunden . . 60
4.6 Antwortzeit der Gemini-Modelle pro Testfrage 60
4.7 Verteilung der Wortfehlerrate für verschiedene Speech-to-Text-Modelle 61
4.8 Real-Time-Factor (RTF) der STT-Modelle in Abhängigkeit von der Audiodauer 62
4.9 Transkriptionszeit der STT-Modelle in Abhängigkeit von der Audiodauer . . . 63
4.10 Generierungsgeschwindigkeit von Text-to-Speech (TTS)-Modellen, gemessen als Real-Time-Factor (RTF) 64
4.11 Skalierbarkeit der TTS-Modelle: Generierungszeit in Abhängigkeit von der Textlänge 65
4.12 Verteilung der Sprechgeschwindigkeit (Zeichen pro Sekunde) für verschiedene TTS-Modelle 65
4.13 Leistungsvergleich von Gesichtsdetektionsmodellen anhand von Precision, Recall und F1-Score 66
4.14 Geschwindigkeitsvergleich von Gesichtsdetektionsmodellen in Frames pro Sekunde (FPS) 67
4.15 Trade-off-Analyse für Gesichtsdetektoren: Genauigkeit (mAP) vs. Detektionsgeschwindigkeit 68
4.16 Precision-Recall-Kurven der evaluierten Gesichtsdetektionsmodelle 69
4.17 Vergleich von Precision und Recall für verschiedene Gesichtserkennungsmodelle 70
4.18 Gesamtleistungsvergleich von Gesichtserkennungsmodellen anhand verschiedener Metriken 71
4.19 Vergleich der Equal Error Rate (EER) für Gesichtserkennungsmodelle 71
4.20 Trade-off-Analyse für Gesichtserkennung: Genauigkeit vs. Inferenzzeit pro Bildpaar 72
4.21 Anwendung der Face Detection und Recognition 73

Tabellenverzeichnis

2.1 Schlüsseldatensätze für egozentrische Vision und ihre Modalitäten 28
2.2 Übersicht über egozentrische KI-Assistenzsysteme und deren Kernfunktionalitäten 29
2.3 Herausforderungen und Fortschritte bei der egozentrischen Videozusammenfassung 30
2.4 Schlüsselherausforderungen und neue Lösungen in der egozentrischen Gesichtserkennung 31

4.1 WER und CER alle getesten Modell 62
4.2 Finale Auswahl an Modellen 70

Literaturverzeichnis

[1] Bernard, "Geschichte der Schnurlostelefone - seit wann telefonieren wir ohne Kabel?" Jul. 2018. [Online]. Available: https://www.expertentesten.de/geschichte-schnurlostelefone-seit-wann-telefonieren-ohne-kabel/

[2] T. Handel, "Bluetooth: Geschichte und Technik." [Online]. Available: https://www.telecom-handel.de/consumer-communications/bluetooth/bluetooth-geschichte-und-technik-885836.html

[3] "Als sich die Virtualität mit der Realität vermischte." [Online]. Available: https://www.barff.de/die-geschichte-der-augmented-reality

[4] "Die Geschichte der (AR) Augmented Reality und ihre Zukunft - VRX," Apr. 2023, running Time: 289 Section: Augmented Reality. [Online]. Available: https://vrx.vr-expert.com/de/die-geschichte-der-ar-und-ihre-zukunft/

[5] I. Hamilton, "Meta Ray-Ban Glasses Pair To Quest 3 For A Preview Of Our Wearable Future," Jul. 2024. [Online]. Available: https://www.uploadvr.com/metas-ray-ban-glasses-quest-3-pair-through-android/

[6] "The Meta Quest 3S Xbox Edition Proves VR Can't Compete Against AR and Smart Glasses," Jun. 2025, section: VR. [Online]. Available: https://uk.pcmag.com/vr-1/158751/the-meta-quest-3s-xbox-edition-proves-vr-cant-compete-against-ar-and-smart-glasses

[7] "Ray-ban meta," page Version ID: 1293656158. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ray-Ban_Meta&oldid=1293656158

[8] R. Carter. Ray-ban meta smart glasses review: Stylish immersion. [Online]. Available: https://www.xrtoday.com/reviews/ray-ban-meta-smart-glasses-review-stylish-immersion/

[9] Ray-ban meta collection. [Online]. Available: https://www.qualcomm.com/products/mobile/snapdragon/xr-vr-ar/xr-vr-ar-device-finder/ray-ban-meta-collection

[10] M. Growcoot. Meta updates smart glasses policy to expand AI data collection. [Online]. Available: https://petapixel.com/2025/05/01/meta-updates-smart-glasses-policy-to-expand-ai-data-collection/

[11] Alex_King_Ax. Is it possible to develop programs for ray-ban meta? [Online]. Available: https://www.reddit.com/r/RayBanStories/comments/1g9p75k/is_it_possible_to_develop_programs_for_rayban_meta/

[12] N. Lambert. Let me use my local LMs on meta ray-bans. [Online]. Available: https://www.interconnects.ai/p/to-meta-ray-ban-local-ai

[13] E. Waisberg, J. Ong, M. Masalkhi, N. Zaman, P. Sarker, and A. Tavakkoli, "Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments," *Eye*, vol. 38, 12 2023.

[14] Deinhard. Welche vorteile bietet python bei data science projekten? [Online]. Available: https://www.it-schulungen.com/wir-ueber-uns/wissensblog/welche-vorteile-bietet-python-bei-data-science-projekten.html

[15] Python-webentwicklung: Vorteile, ist python gut für frontend und backend. [Online]. Available: https://gpsolutions.com/de/blog/softwareentwicklung/python-fuer-die-webentwicklung/

[16] TU Dresden, "Python in der lehre — institut für regelungs- und steuerungstheorie," https://tu-dresden.de/ing/elektrotechnik/rst/ressourcen/dateien/lehre/dateien/python-in-der-lehre?lang=de, 2022, zugriff am: Juli 8, 2025.

[17] Eine kurze einführung in programmiersprachen: Python – die programmiersprache für data engineering und KI. [Online]. Available: https://www.adesso.de/de/news/blog/eine-kurze-einfuehrung-in-programmiersprachen-python-die-programmiersprache-fuer-data-engineering-und-ki.jsp

[18] Selenium-tutorial: So nutzen sie das cross-browser-testing-framework. [Online]. Available: https://www.ionos.de/digitalguide/websites/web-entwicklung/selenium-webdriver-tutorial/

[19] Thunderbit, "Die 15 besten browser-automatisierungstools 2025," https://thunderbit.com/de/browser-automatisierungstools/, 2024, zugriff am: Juli 8, 2025.

[20] G. Lee. Selenium load testing erklärt: WebDriver & java. [Online]. Available: https://www.loadview-testing.com/de/blog/selenium-load-testing-erklaert-webdriver-java/

[21] Selenium WebDriver: Step-by-step tutorial with examples. Section: Selenium. [Online]. Available: https://testgrid.io/blog/selenium-webdriver/

[22] headmaster. Selenium WebDriver - rahmenarchitektur - ittester.sk. [Online]. Available: https://ittester.sk/de/automatisiertes-testen/selenium-tutorial-de/selenium-webdriver-architektur/

[23] Naveen AutomationLabs, "What is selenium webdriver architecture?" https://www.youtube.com/watch?v=52K1g93i_UU, 2017, zugriff am: Juli 8, 2025.

[24] AbdelRhman_Sabry. Scraping facebook in 2025: Combining selenium and BeautifulSoup for effective data extraction. [Online]. Available: https:

//medium.com/@AbdelRhman_Sabry/scraping-facebook-in-2025-combining-selenium-and-beautifulsoup-for-effective-data-extraction-95bc8d705889

[25] Wie kann man python-scraping mit selenium erstellen? [Online]. Available: https://www.octoparse.de/blog/wie-erstellt-man-python-scraper-mit-selenium

[26] A. Kozinskiy. Facebook API tutorial. learn about facebook graph. [Online]. Available: https://elfsight.com/blog/how-to-get-and-use-facebook-api/

[27] Facebook messenger for business: Eine komplettanleitung. [Online]. Available: https://www.zendesk.de/service/messaging/facebook-messenger-business/

[28] Stack Overflow User, "Facebook messenger api pages_messaging permission," https://stackoverflow.com/questions/38006456/facebook-messenger-api-pages-messaging-permission, 2016, zugriff am: Juli 8, 2025.

[29] ——, "Sending private messages through facebook api," https://stackoverflow.com/questions/4414392/sending-private-messages-through-facebook-api, 2010, zugriff am: Juli 8, 2025.

[30] S. Z. Collecting public data from facebook using selenium and beautiful soup. [Online]. Available: https://medium.com/nerd-for-tech/collecting-public-data-from-facebook-using-selenium-and-beautiful-soup-f0f918971000

[31] M. Sha, "Web scraping facebook with selenium - automated bot," https://www.youtube.com/watch?v=S3w9x_aJ1F0, 2021, zugriff am: Juli 8, 2025.

[32] Stack Overflow User, "Getting to my profile in facebook for a self web scrapping project using selenium," https://stackoverflow.com/questions/71714930/getting-to-my-profile-in-facebook-for-a-self-web-scrapping-project-using-seleni, 2022, zugriff am: Juli 8, 2025.

[33] ultrafunkamsterdam. Scraping facebook and tiktok using selenium. · ultrafunkamsterdam/undetected-chromedriver · discussion #2180. [Online]. Available: https://github.com/ultrafunkamsterdam/undetected-chromedriver/discussions/2180

[34] "Language model," page Version ID: 1297542053. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Language_model&oldid=1297542053

[35] Was ist LLM? – große sprachmodelle erklärt – AWS. [Online]. Available: https://aws.amazon.com/de/what-is/large-language-model/

[36] meta-llama/llama-3.3-70b-instruct · hugging face. [Online]. Available: https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

[37] What are large language models (LLMs)? | IBM. [Online]. Available: https://www.ibm.com/think/topics/large-language-models

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need." [Online]. Available: http://arxiv.org/abs/1706.03762

[39] What is a transformer model? | IBM. [Online]. Available: https://www.ibm.com/think/topics/transformer-model

[40] S. Raschka. Understanding and coding the self-attention mechanism of large language models from scratch. [Online]. Available: https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

[41] What is self-attention? | IBM. [Online]. Available: https://www.ibm.com/think/topics/self-attention

[42] What is an attention mechanism? | IBM. [Online]. Available: https://www.ibm.com/think/topics/attention-mechanism

[43] "11.7. The Transformer Architecture ; Dive into Deep Learning 1.0.3 documentation — d2l.ai," https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html, [Accessed 24-08-2025].

[44] A. M. PhD, "Google's Gemini 2.5 Technical Report — a new paradigm of autonomous, multimodal systems," Jul. 2025. [Online]. Available: https://medium.com/@adnanmasood/googles-gemini-2-5-technical-report-a-new-paradigm-of-autonomous-multimodal-systems-44e37c2d4358

[45] M. Verma, S. Bhambri, and S. Kambhampati, "On the brittle foundations of ReAct prompting for agentic large language models." [Online]. Available: http://arxiv.org/abs/2405.13966

[46] "Die Macht der Transformer-Architektur - Bavest Blog | Bavest." [Online]. Available: https://www.bavest.co/de/post/unlocking-the-power-of-transformers

[47] "Gemini 1.5: Google's Generative AI Model with Mixture of Experts Architecture." [Online]. Available: https://encord.com/blog/google-gemini-1-5-generative-ai-model-with-mixture-of-experts/

[48] "Gemini (language model)," Aug. 2025, page Version ID: 1305779202. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Gemini_(language_model)&oldid=1305779202

[49] B. Dickson, "What to know about Gemini 2.5 Deep Think - TechTalks," Aug. 2025. [Online]. Available: https://bdtechtalks.com/2025/08/03/gemini-2-5-deep-think/

[50] "Mixture of experts," Jul. 2025, page Version ID: 1300103120. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mixture_of_experts&oldid=1300103120

[51] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, S. Mariooryad, Y. Ding, X. Geng, F. Alcober, R. Frostig, M. Omernick, L. Walker, C. Paduraru, C. Sorokin, A. Tacchetti, C. Gaffney, S. Daruki, O. Sercinoglu, Z. Gleicher, J. Love, P. Voigtlaender, R. Jain, G. Surita, K. Mohamed, R. Blevins, J. Ahn, T. Zhu, K. Kawintiranon, O. Firat, Y. Gu, Y. Zhang, M. Rahtz, M. Faruqui, N. Clay, J. Gilmer, J. Co-Reyes, I. Penchev, R. Zhu, N. Morioka, K. Hui, K. Haridasan, V. Campos, M. Mahdieh, M. Guo, S. Hassan, K. Kilgour, A. Vezer,

H.-T. Cheng, R. de Liedekerke, S. Goyal, P. Barham, D. Strouse, S. Noury, J. Adler, M. Sundararajan, S. Vikram, D. Lepikhin, M. Paganini, X. Garcia, F. Yang, D. Valter, M. Trebacz, K. Vodrahalli, C. Asawaroengchai, R. Ring, N. Kalb, L. B. Soares, S. Brahma, D. Steiner, T. Yu, F. Mentzer, A. He, L. Gonzalez, B. Xu, R. L. Kaufman, L. E. Shafey, J. Oh, T. Hennigan, G. van den Driessche, S. Odoom, M. Lucic, B. Roelofs, S. Lall, A. Marathe, B. Chan, S. Ontanon, L. He, D. Teplyashin, J. Lai, P. Crone, B. Damoc, L. Ho, S. Riedel, K. Lenc, C.-K. Yeh, A. Chowdhery, Y. Xu, M. Kazemi, E. Amid, A. Petrushkina, K. Swersky, A. Khodaei, G. Chen, C. Larkin, M. Pinto, G. Yan, A. P. Badia, P. Patil, S. Hansen, D. Orr, S. M. R. Arnold, J. Grimstad, A. Dai, S. Douglas, R. Sinha, V. Yadav, X. Chen, E. Gribovskaya, J. Austin, J. Zhao, K. Patel, P. Komarek, S. Austin, S. Borgeaud, L. Friso, A. Goyal, B. Caine, K. Cao, D.-W. Chung, M. Lamm, G. Barth-Maron, T. Kagohara, K. Olszewska, M. Chen, K. Shivakumar, R. Agarwal, H. Godhia, R. Rajwar, J. Snaider, X. Dotiwalla, Y. Liu, A. Barua, V. Ungureanu, Y. Zhang, B.-O. Batsaikhan, M. Wirth, J. Qin, I. Danihelka, T. Doshi, M. Chadwick, J. Chen, S. Jain, Q. Le, A. Kar, M. Gurumurthy, C. Li, R. Sang, F. Liu, L. Lamprou, R. Munoz, N. Lintz, H. Mehta, H. Howard, M. Reynolds, L. Aroyo, Q. Wang, L. Blanco, A. Cassirer, J. Griffith, D. Das, S. Lee, J. Sygnowski, Z. Fisher, J. Besley, R. Powell, Z. Ahmed, D. Paulus, D. Reitter, Z. Borsos, R. Joshi, A. Pope, S. Hand, V. Selo, V. Jain, N. Sethi, M. Goel, T. Makino, R. May, Z. Yang, J. Schalkwyk, C. Butterfield, A. Hauth, A. Goldin, W. Hawkins, E. Senter, S. Brin, O. Woodman, M. Ritter, E. Noland, M. Giang, V. Bolina, L. Lee, T. Blyth, I. Mackinnon, M. Reid, O. Sarvana, D. Silver, A. Chen, L. Wang, L. Maggiore, O. Chang, N. Attaluri, G. Thornton, C.-C. Chiu, O. Bunyan, N. Levine, T. Chung, E. Eltyshev, X. Si, T. Lillicrap, D. Brady, V. Aggarwal, B. Wu, Y. Xu, R. McIlroy, K. Badola, P. Sandhu, E. Moreira, W. Stokowiec, R. Hemsley, D. Li, A. Tudor, P. Shyam, E. Rahimtoroghi, S. Haykal, P. Sprechmann, X. Zhou, D. Mincu, Y. Li, R. Addanki, K. Krishna, X. Wu, A. Frechette, M. Eyal, A. Dafoe, D. Lacey, J. Whang, T. Avrahami, Y. Zhang, E. Taropa, H. Lin, D. Toyama, E. Rutherford, M. Sano, H. Choe, A. Tomala, C. Safranek-Shrader, N. Kassner, M. Pajarskas, M. Harvey, S. Sechrist, M. Fortunato, C. Lyu, G. Elsayed, C. Kuang, J. Lottes, E. Chu, C. Jia, C.-W. Chen, P. Humphreys, K. Baumli, C. Tao, R. Samuel, C. N. dos Santos, A. Andreassen, N. Rakićević, D. Grewe, A. Kumar, S. Winkler, J. Caton, A. Brock, S. Dalmia, H. Sheahan, I. Barr, Y. Miao, P. Natsev, J. Devlin, F. Behbahani, F. Prost, Y. Sun, A. Myaskovsky, T. S. Pillai, D. Hurt, A. Lazaridou, X. Xiong, C. Zheng, F. Pardo, X. Li, D. Horgan, J. Stanton, M. Ambar, F. Xia, A. Lince, M. Wang, B. Mustafa, A. Webson, H. Lee, R. Anil, M. Wicke, T. Dozat, A. Sinha, E. Piqueras, E. Dabir, S. Upadhyay, A. Boral, L. A. Hendricks, C. Fry, J. Djolonga, Y. Su, J. Walker, J. Labanowski, R. Huang, V. Misra, J. Chen, R. Skerry-Ryan, A. Singh, S. Rijhwani, D. Yu, A. Castro-Ros, B. Changpinyo, R. Datta, S. Bagri, A. M. Hrafnkelsson, M. Maggioni, D. Zheng, Y. Sulsky, S. Hou, T. L. Paine, A. Yang, J. Riesa, D. Rogozinska, D. Marcus, D. E. Badawy, Q. Zhang, L. Wang, H. Miller, J. Greer, L. L. Sjos, A. Nova, H. Zen, R. Chaabouni, M. Rosca, J. Jiang, C. Chen, R. Liu, T. Sainath, M. Krikun, A. Polozov, J.-B. Lespiau, J. Newlan, Z. Cankara, S. Kwak, Y. Xu, P. Chen, A. Coenen, C. Meyer, K. Tsihlas, A. Ma, J. Gottweis, J. Xing, C. Gu, J. Miao, C. Frank, Z. Cankara, S. Ganapathy,

I. Dasgupta, S. Hughes-Fitt, H. Chen, D. Reid, K. Rong, H. Fan, J. van Amersfoort, V. Zhuang, A. Cohen, S. S. Gu, A. Mohananey, A. Ilic, T. Tobin, J. Wieting, A. Bortsova, P. Thacker, E. Wang, E. Caveness, J. Chiu, E. Sezener, A. Kaskasoli, S. Baker, K. Millican, M. Elhawaty, K. Aisopos, C. Lebsack, N. Byrd, H. Dai, W. Jia, M. Wiethoff, E. Davoodi, A. Weston, L. Yagati, A. Ahuja, I. Gao, G. Pundak, S. Zhang, M. Azzam, K. C. Sim, S. Caelles, J. Keeling, A. Sharma, A. Swing, Y. Li, C. Liu, C. G. Bostock, Y. Bansal, Z. Nado, A. Anand, J. Lipschultz, A. Karmarkar, L. Proleev, A. Ittycheriah, S. H. Yeganeh, G. Polovets, A. Faust, J. Sun, A. Rrustemi, P. Li, R. Shivanna, J. Liu, C. Welty, F. Lebron, A. Baddepudi, S. Krause, E. Parisotto, R. Soricut, Z. Xu, D. Bloxwich, M. Johnson, B. Neyshabur, J. Mao-Jones, R. Wang, V. Ramasesh, Z. Abbas, A. Guez, C. Segal, D. D. Nguyen, J. Svensson, L. Hou, S. York, K. Milan, S. Bridgers, W. Gworek, M. Tagliasacchi, J. Lee-Thorp, M. Chang, A. Guseynov, A. J. Hartman, M. Kwong, R. Zhao, S. Kashem, E. Cole, A. Miech, R. Tanburn, M. Phuong, F. Pavetic, S. Cevey, R. Comanescu, R. Ives, S. Yang, C. Du, B. Li, Z. Zhang, M. Iinuma, C. H. Hu, A. Roy, S. Bijwadia, Z. Zhu, D. Martins, R. Saputro, A. Gergely, S. Zheng, D. Jia, I. Antonoglou, A. Sadovsky, S. Gu, Y. Bi, A. Andreev, S. Samangooei, M. Khan, T. Kocisky, A. Filos, C. Kumar, C. Bishop, A. Yu, S. Hodkinson, S. Mittal, P. Shah, A. Moufarek, Y. Cheng, A. Bloniarz, J. Lee, P. Pejman, P. Michel, S. Spencer, V. Feinberg, X. Xiong, N. Savinov, C. Smith, S. Shakeri, D. Tran, M. Chesus, B. Bohnet, G. Tucker, T. von Glehn, C. Muir, Y. Mao, H. Kazawa, A. Slone, K. Soparkar, D. Shrivastava, J. Cobon-Kerr, M. Sharman, J. Pavagadhi, C. Araya, K. Misiunas, N. Ghelani, M. Laskin, D. Barker, Q. Li, A. Briukhov, N. Houlsby, M. Glaese, B. Lakshminarayanan, N. Schucher, Y. Tang, E. Collins, H. Lim, F. Feng, A. Recasens, G. Lai, A. Magni, N. D. Cao, A. Siddhant, Z. Ashwood, J. Orbay, M. Dehghani, J. Brennan, Y. He, K. Xu, Y. Gao, C. Saroufim, J. Molloy, X. Wu, S. Arnold, S. Chang, J. Schrittwieser, E. Buchatskaya, S. Radpour, M. Polacek, S. Giordano, A. Bapna, S. Tokumine, V. Hellendoorn, T. Sottiaux, S. Cogan, A. Severyn, M. Saleh, S. Thakoor, L. Shefey, S. Qiao, M. Gaba, S. yiin Chang, C. Swanson, B. Zhang, B. Lee, P. K. Rubenstein, G. Song, T. Kwiatkowski, A. Koop, A. Kannan, D. Kao, P. Schuh, A. Stjerngren, G. Ghiasi, G. Gibson, L. Vilnis, Y. Yuan, F. T. Ferreira, A. Kamath, T. Klimenko, K. Franko, K. Xiao, I. Bhattacharya, M. Patel, R. Wang, A. Morris, R. Strudel, V. Sharma, P. Choy, S. H. Hashemi, J. Landon, M. Finkelstein, P. Jhakra, J. Frye, M. Barnes, M. Mauger, D. Daun, K. Baatarsukh, M. Tung, W. Farhan, H. Michalewski, F. Viola, F. de Chaumont Quitry, C. L. Lan, T. Hudson, Q. Wang, F. Fischer, I. Zheng, E. White, A. Dragan, J. baptiste Alayrac, E. Ni, A. Pritzel, A. Iwanicki, M. Isard, A. Bulanova, L. Zilka, E. Dyer, D. Sachan, S. Srinivasan, H. Muckenhirn, H. Cai, A. Mandhane, M. Tariq, J. W. Rae, G. Wang, K. Ayoub, N. FitzGerald, Y. Zhao, W. Han, C. Alberti, D. Garrette, K. Krishnakumar, M. Gimenez, A. Levskaya, D. Sohn, J. Matak, I. Iturrate, M. B. Chang, J. Xiang, Y. Cao, N. Ranka, G. Brown, A. Hutter, V. Mirrokni, N. Chen, K. Yao, Z. Egyed, F. Galilee, T. Liechty, P. Kallakuri, E. Palmer, S. Ghemawat, J. Liu, D. Tao, C. Thornton, T. Green, M. Jasarevic, S. Lin, V. Cotruta, Y.-X. Tan, N. Fiedel, H. Yu, E. Chi, A. Neitz, J. Heitkaemper, A. Sinha, D. Zhou, Y. Sun, C. Kaed, B. Hulse, S. Mishra, M. Georgaki, S. Kudugunta, C. Farabet, I. Shafran, D. Vlasic, A. Tsitsulin, R. Ananthanarayanan, A. Carin, G. Su, P. Sun, S. V,

G. Carvajal, J. Broder, I. Comsa, A. Repina, W. Wong, W. W. Chen, P. Hawkins, E. Filonov, L. Loher, C. Hirnschall, W. Wang, J. Ye, A. Burns, H. Cate, D. G. Wright, F. Piccinini, L. Zhang, C.-C. Lin, I. Gog, Y. Kulizhskaya, A. Sreevatsa, S. Song, L. C. Cobo, A. Iyer, C. Tekur, G. Garrido, Z. Xiao, R. Kemp, H. S. Zheng, H. Li, A. Agarwal, C. Ngani, K. Goshvadi, R. Santamaria-Fernandez, W. Fica, X. Chen, C. Gorgolewski, S. Sun, R. Garg, X. Ye, S. M. A. Eslami, N. Hua, J. Simon, P. Joshi, Y. Kim, I. Tenney, S. Potluri, L. N. Thiet, Q. Yuan, F. Luisier, A. Chronopoulou, S. Scellato, P. Srinivasan, M. Chen, V. Koverkathu, V. Dalibard, Y. Xu, B. Saeta, K. Anderson, T. Sellam, N. Fernando, F. Huot, J. Jung, M. Varadarajan, M. Quinn, A. Raul, M. Le, R. Habalov, J. Clark, K. Jalan, K. Bullard, A. Singhal, T. Luong, B. Wang, S. Rajayogam, J. Eisenschlos, J. Jia, D. Finchelstein, A. Yakubovich, D. Balle, M. Fink, S. Agarwal, J. Li, D. Dvijotham, S. Pal, K. Kang, J. Konzelmann, J. Beattie, O. Dousse, D. Wu, R. Crocker, C. Elkind, S. R. Jonnalagadda, J. Lee, D. Holtmann-Rice, K. Kallarackal, R. Liu, D. Vnukov, N. Vats, L. Invernizzi, M. Jafari, H. Zhou, L. Taylor, J. Prendki, M. Wu, T. Eccles, T. Liu, K. Kopparapu, F. Beaufays, C. Angermueller, A. Marzoca, S. Sarcar, H. Dib, J. Stanway, F. Perbet, N. Trdin, R. Sterneck, A. Khorlin, D. Li, X. Wu, S. Goenka, D. Madras, S. Goldshtein, W. Gierke, T. Zhou, Y. Liu, Y. Liang, A. White, Y. Li, S. Singh, S. Bahargam, M. Epstein, S. Basu, L. Lao, A. Ozturel, C. Crous, A. Zhai, H. Lu, Z. Tung, N. Gaur, A. Walton, L. Dixon, M. Zhang, A. Globerson, G. Uy, A. Bolt, O. Wiles, M. Nasr, I. Shumailov, M. Selvi, F. Piccinno, R. Aguilar, S. McCarthy, M. Khalman, M. Shukla, V. Galic, J. Carpenter, K. Villela, H. Zhang, H. Richardson, J. Martens, M. Bosnjak, S. R. Belle, J. Seibert, M. Alnahlawi, B. McWilliams, S. Singh, A. Louis, W. Ding, D. Popovici, L. Simicich, L. Knight, P. Mehta, N. Gupta, C. Shi, S. Fatehi, J. Mitrovic, A. Grills, J. Pagadora, T. Munkhdalai, D. Petrova, D. Eisenbud, Z. Zhang, D. Yates, B. Mittal, N. Tripuraneni, Y. Assael, T. Brovelli, P. Jain, M. Velimirovic, C. Akbulut, J. Mu, W. Macherey, R. Kumar, J. Xu, H. Qureshi, G. Comanici, J. Wiesner, Z. Gong, A. Ruddock, M. Bauer, N. Felt, A. GP, A. Arnab, D. Zelle, J. Rothfuss, B. Rosgen, A. Shenoy, B. Seybold, X. Li, J. Mudigonda, G. Erdogan, J. Xia, J. Simsa, A. Michi, Y. Yao, C. Yew, S. Kan, I. Caswell, C. Radebaugh, A. Elisseeff, P. Valenzuela, K. McKinney, K. Paterson, A. Cui, E. Latorre-Chimoto, S. Kim, W. Zeng, K. Durden, P. Ponnapalli, T. Sosea, C. A. Choquette-Choo, J. Manyika, B. Robenek, H. Vashisht, S. Pereira, H. Lam, M. Velic, D. Owusu-Afriyie, K. Lee, T. Bolukbasi, A. Parrish, S. Lu, J. Park, B. Venkatraman, A. Talbert, L. Rosique, Y. Cheng, A. Sozanschi, A. Paszke, P. Kumar, J. Austin, L. Li, K. Salama, B. Perz, W. Kim, N. Dukkipati, A. Baryshnikov, C. Kaplanis, X. Sheng, Y. Chervonyi, C. Unlu, D. de Las Casas, H. Askham, K. Tunyasuvunakool, F. Gimeno, S. Poder, C. Kwak, M. Miecnikowski, V. Mirrokni, A. Dimitriev, A. Parisi, D. Liu, T. Tsai, T. Shevlane, C. Kouridi, D. Garmon, A. Goedeckemeyer, A. R. Brown, A. Vijayakumar, A. Elqursh, S. Jazayeri, J. Huang, S. M. Carthy, J. Hoover, L. Kim, S. Kumar, W. Chen, C. Biles, G. Bingham, E. Rosen, L. Wang, Q. Tan, D. Engel, F. Pongetti, D. de Cesare, D. Hwang, L. Yu, J. Pullman, S. Narayanan, K. Levin, S. Gopal, M. Li, A. Aharoni, T. Trinh, J. Lo, N. Casagrande, R. Vij, L. Matthey, B. Ramadhana, A. Matthews, C. Carey, M. Johnson, K. Goranova, R. Shah, S. Ashraf, K. Dasgupta, R. Larsen, Y. Wang, M. R. Vuyyuru, C. Jiang, J. Ijazi, K. Osawa, C. Smith, R. S. Boppana, T. Bilal,

Y. Koizumi, Y. Xu, Y. Altun, N. Shabat, B. Bariach, A. Korchemniy, K. Choo, O. Ronneberger, C. Iwuanyanwu, S. Zhao, D. Soergel, C.-J. Hsieh, I. Cai, S. Iqbal, M. Sundermeyer, Z. Chen, E. Bursztein, C. Malaviya, F. Biadsy, P. Shroff, I. Dhillon, T. Latkar, C. Dyer, H. Forbes, M. Nicosia, V. Nikolaev, S. Greene, M. Georgiev, P. Wang, N. Martin, H. Sedghi, J. Zhang, P. Banzal, D. Fritz, V. Rao, X. Wang, J. Zhang, V. Patraucean, D. Du, I. Mordatch, I. Jurin, L. Liu, A. Dubey, A. Mohan, J. Nowakowski, V.-D. Ion, N. Wei, R. Tojo, M. A. Raad, D. A. Hudson, V. Keshava, S. Agrawal, K. Ramirez, Z. Wu, H. Nguyen, J. Liu, M. Sewak, B. Petrini, D. Choi, I. Philips, Z. Wang, I. Bica, A. Garg, J. Wilkiewicz, P. Agrawal, X. Li, D. Guo, E. Xue, N. Shaik, A. Leach, S. M. Khan, J. Wiesinger, S. Jerome, A. Chakladar, A. W. Wang, T. Ornduff, F. Abu, A. Ghaffarkhah, M. Wainwright, M. Cortes, F. Liu, J. Maynez, A. Terzis, P. Samangouei, R. Mansour, T. Kepa, F.-X. Aubet, A. Algymr, D. Banica, A. Weisz, A. Orban, A. Senges, E. Andrejczuk, M. Geller, N. D. Santo, V. Anklin, M. A. Merey, M. Baeuml, T. Strohman, J. Bai, S. Petrov, Y. Wu, D. Hassabis, K. Kavukcuoglu, J. Dean, and O. Vinyals, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," 2024. [Online]. Available: https://arxiv.org/abs/2403.05530

[52] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," Jun. 2020, arXiv:2006.16668 [cs]. [Online]. Available: http://arxiv.org/abs/2006.16668

[53] K. Dhanakotti, "Unlocking Efficiency and Scale: The Mixture of Experts (MoE) and Sparse MoE (SMoE) Architectures...," Aug. 2024. [Online]. Available: https://dkaarthick.medium.com/unlocking-efficiency-and-scale-the-mixture-of-experts-moe-and-sparse-moe-smoe-architectures-676fffaac2db

[54] "Mixture of Experts Explained," Feb. 2025. [Online]. Available: https://huggingface.co/blog/moe

[55] M. Grootendorst, "A Visual Guide to Mixture of Experts (MoE)," Feb. 2024. [Online]. Available: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mixture-of-experts

[56] L. Lango, "The Little-Known Chip Powering the Very Future of AI," Jul. 2025. [Online]. Available: https://investorplace.com/hypergrowthinvesting/2025/07/how-googles-tpu-is-powering-the-very-future-of-ai/

[57] "Trillium TPU is GA." [Online]. Available: https://cloud.google.com/blog/products/compute/trillium-tpu-is-ga

[58] "Gemini 2.5: Our most intelligent AI model," Mar. 2025. [Online]. Available: https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/

[59] Google text to speech for AI voice generator: review, features & use cases. [Online]. Available: https://google-text-to-speech.en.softonic.com/web-apps

[60] "Gemini 2.5 Flash-Lite is now stable and generally available- Google Developers Blog." [Online]. Available: https://developers.googleblog.com/en/gemini-25-flash-lite-is-now-stable-and-generally-available/?utm_source=chatgpt.com

[61] "Gemini 2.5 Flash-Lite | Generative AI on Vertex AI." [Online]. Available: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash-lite?hl=de

[62] "Vertex AI – Preise | Generative AI on Vertex AI." [Online]. Available: https://cloud.google.com/vertex-ai/generative-ai/pricing?hl=de

[63] Speech-to-text - ausführliche erklärung aus dem KI-lexikon. [Online]. Available: https://www.dogado.de/ki-lexikon/speech-to-text

[64] Was ist sprache-zu-text? – sprache-zu-text-transkription erklärt – AWS. [Online]. Available: https://aws.amazon.com/de/what-is/speech-to-text/

[65] "alphacep/vosk-api," Aug. 2025, original-date: 2019-09-03T17:48:42Z. [Online]. Available: https://github.com/alphacep/vosk-api

[66] "Offline speech recognition on Android with VOSK." [Online]. Available: https://alphacephei.com/vosk/android

[67] "Kaldi: 'Chain' models." [Online]. Available: https://kaldi-asr.org/doc/chain.html?utm_source=chatgpt.com

[68] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks," in *Interspeech 2018*. ISCA, Sep. 2018, pp. 3743–3747. [Online]. Available: https://www.isca-archive.org/interspeech_2018/povey18_interspeech.html

[69] J. Zhao, G. Shi, G.-B. Wang, and W.-Q. Zhang, "The THUEE Systems for the IARPA OpenASR20 Evaluation."

[70] J. Ma, F. Keith, T. Ng, M.-H. Siu, and O. Kimball, "Improving Deliverable Speech-to-Text Systems with Multilingual Knowledge Transfer," in *Interspeech 2017*. ISCA, Aug. 2017, pp. 127–131. [Online]. Available: https://www.isca-archive.org/interspeech_2017/ma17_interspeech.html

[71] "VOSK language model adaptation." [Online]. Available: https://alphacephei.com/vosk/lm

[72] "Wir stellen vor: Whisper," Apr. 2022. [Online]. Available: https://openai.com/de-DE/index/whisper/

[73] "Unveiling the Power of Whisper AI: Architecture and Practical Implementations – Developer Blog." [Online]. Available: https://via-internet.de/blog/2023/12/16/unveiling-the-power-of-whisper-ai-architecture-and-practical-implementations/

[74] "Whisper Variants Comparison: What Are Their Features And How To Implement Them?" Nov. 2024. [Online]. Available: https://app.daily.dev/posts/whisper-variants-comparison-what-are-their-features-and-how-to-implement-them--x64gtc5fe

[75] A. Hasan, "Decoding Whisper: An In-Depth Look at its Architecture and Transcription Process," Jan. 2024. [Online]. Available: https://amgadhasan.substack.com/p/whisper-how-to-create-robust-asr-46b

[76] "Speech-to-Text with Faster-Whisper: The High-Speed Alternative to OpenAI Whisper." [Online]. Available: https://mysoly.nl/speech-to-text-with-faster-whisper-the-high-speed-alternative-to-openai-whisper/

[77] O. Okoye, "Whisper: Functionality and Finetuning," Nov. 2024. [Online]. Available: https://medium.com/@okezieowen/whisper-functionality-and-finetuning-ba7f9444f55a

[78] D. Cochard, "Whisper : Speech recognition model capable of recognizing 99 languages," Nov. 2023. [Online]. Available: https://medium.com/axinc-ai/whisper-speech-recognition-model-capable-of-recognizing-99-languages-5b5cf0197c16

[79] A. Hasan, "Demystifying OpenAI's new Whisper Turbo," Oct. 2024. [Online]. Available: https://amgadhasan.substack.com/p/demystifying-openais-new-whisper

[80] "openai/whisper," Aug. 2025, original-date: 2022-09-16T20:02:54Z. [Online]. Available: https://github.com/openai/whisper

[81] Y. Shizuya, "Automatic Speech Recognition -Whisper Variants Comparison: What Are Their Features And How To...," Nov. 2024. [Online]. Available: https://pub.towardsai.net/whisper-variants-comparison-what-are-their-features-and-how-to-implement-them-c3eb07b6eb95

[82] "Showdown of Whisper Variants – Quids." [Online]. Available: https://quids.tech/blog/showdown-of-whisper-variants/

[83] "Which Whisper Model Should I Choose?" Mar. 2025. [Online]. Available: https://whisper-api.com/blog/models/

[84] "SYSTRAN/faster-whisper," Aug. 2025, original-date: 2023-02-11T09:17:27Z. [Online]. Available: https://github.com/SYSTRAN/faster-whisper

[85] "OpenNMT/CTranslate2," Aug. 2025, original-date: 2019-09-23T08:10:42Z. [Online]. Available: https://github.com/OpenNMT/CTranslate2

[86] guillaumekln, "[P] CTranslate2: an efficient inference engine for Transformer models," May 2022. [Online]. Available: https://www.reddit.com/r/MachineLearning/comments/uw1o0w/p_ctranslate2_an_efficient_inference_engine_for/

[87] Trelis Research, "Fine tune and Serve Faster Whisper Turbo," Oct. 2024. [Online]. Available: https://www.youtube.com/watch?v=qXtPPgujufI

[88] "'turbo' model release · openai/whisper · Discussion #2363." [Online]. Available: https://github.com/openai/whisper/discussions/2363

[89] GroqDocs, "Whisper large v3 turbo," https://groq.com/docs/groqcloud/speech-to-text/whisper-large-v3-turbo/, Juli 2025, zugriff am: Juli 8, 2025.

[90] openai/whisper-large-v3-turbo - demo - DeepInfra. [Online]. Available: https://deepinfra.com/openai/whisper-large-v3-turbo

[91] "openai/whisper-large-v3-turbo · Hugging Face." [Online]. Available: https://huggingface.co/openai/whisper-large-v3-turbo

[92] B. Marie, "Whisper Large V3 Turbo: As Good as Large V2 but 6x Faster," Oct. 2024. [Online]. Available: https://medium.com/@bnjmn_marie/whisper-large-v3-turbo-as-good-as-large-v2-but-6x-faster-97f0803fa933

[93] M.-L. Stadler. Text-to-speech (TTS). [Online]. Available: https://mindsquare.de/knowhow/text-to-speech/

[94] B. d.-A. f. d. Kommunikation. Text-to-speech - was ist das? erklärung und information. [Online]. Available: https://botschaft.digital/glossar/text-to-speech

[95] "Sprachsynthese," page Version ID: 256477793. [Online]. Available: https://de.wikipedia.org/w/index.php?title=Sprachsynthese&oldid=256477793

[96] Speechify, "Text-to-speech (tts). alles, was sie wissen müssen!" https://speechify.com/de/blog/text-to-speech/, Juni 2022, zugriff am: Juli 8, 2025.

[97] L. AI. AI voice generator: Realistic text to speech & voice cloning. [Online]. Available: https://listnr.ai/

[98] Text-to-speech AI: Lifelike speech synthesis. [Online]. Available: https://cloud.google.com/text-to-speech

[99] Everything to know about google cloud text to speech API. Section: API. [Online]. Available: https://speechify.com/blog/google-text-to-speech-api/

[100] "Tacotron 2 – PyTorch." [Online]. Available: https://pytorch.org/hub/nvidia_deeplearningexamples_tacotron2/

[101] "Thorsten-Voice/Tacotron2-DDC · Hugging Face." [Online]. Available: https://huggingface.co/Thorsten-Voice/Tacotron2-DDC

[102] K. Chakrabarti, "Text to Speech with Tacotron2 -Part 5 -Implementation," Jun. 2025. [Online]. Available: https://medium.com/@koyelac/text-to-speech-with-tacotron2-part-5-implementation-b9811154387d

[103] "Sesame CSM 1B - AI Speech Generation Model," Mar. 2025. [Online]. Available: https://csm1b.com/

[104] M. Gupta, "Sesame CSM-1b: More Than TTS, ElevenLabs Free Alternative," Mar. 2025. [Online]. Available: https://medium.com/data-science-in-your-pocket/sesame-csm-1b-more-than-tts-elevenlabs-free-alternative-b1a3bdffcea2

[105] "An Overview of Sesame's Conversational Speech Model | DigitalOcean." [Online]. Available: https://www.digitalocean.com/community/tutorials/sesame-csm

[106] A. Łańcucki, "FastPitch: Parallel Text-to-speech with Pitch Prediction," Feb. 2021, arXiv:2006.06873 [eess]. [Online]. Available: http://arxiv.org/abs/2006.06873

[107] "fastpitch: A Deep Learning Framework for Text-to-Speech Synthesis." [Online]. Available: https://serp.ai/posts/fastpitch/

[108] "Speech Synthesis English FastPitch | NVIDIA NGC." [Online]. Available: https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/speechsynthesis_english_fastpitch

[109] Z. Song, "szmmm/ESPNet2," Nov. 2020, original-date: 2020-11-05T20:40:10Z. [Online]. Available: https://github.com/szmmm/ESPNet2

[110] T. Desot, F. Portet, and M. Vacher, "End-to-End Spoken Language Understanding: Performance analyses of a voice command task in a low resource setting," *Computer Speech & Language*, vol. 75, p. 101369, Sep. 2022, arXiv:2207.08179 [cs]. [Online]. Available: http://arxiv.org/abs/2207.08179

[111] "espnet/espnet," Aug. 2025, original-date: 2017-12-13T00:45:11Z. [Online]. Available: https://github.com/espnet/espnet

[112] "espnet/kan-bayashi_ljspeech_vits · Hugging Face." [Online]. Available: https://huggingface.co/espnet/kan-bayashi_ljspeech_vits

[113] "Introduction of ESPnet, end-to-end speech processing toolkit: new features, broadened applications, performance improvements, and future challenges," 2000.

[114] "pyttsx3: Text to Speech (TTS) library for Python 3. Works without internet connection or delay. Supports multiple TTS engines, including Sapi5, nsss, and espeak." [Online]. Available: https://github.com/nateshmbhat/pyttsx3

[115] "Using pyttsx3 — pyttsx3 2.6 documentation." [Online]. Available: https://pyttsx3.readthedocs.io/en/latest/engine.html

[116] "Text-to-Speech with Python: A Beginner's Guide to PYTTSX3," Dec. 2024. [Online]. Available: https://dev.to/mr_nova/text-to-speech-with-python-a-beginners-guide-to-pyttsx3-2pie

[117] "Supported synthesizers — pyttsx3 2.6 documentation." [Online]. Available: https://pyttsx3.readthedocs.io/en/latest/support.html

[118] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, pp. I–511, 2001.

[119] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," in *IEEE Signal Processing Letters*, vol. 23, no. 10. IEEE, 2016, pp. 1499–1503.

[120] A. F. Abate, F. Naccari, G. Naccari, and R. Naccari, "Face detection and recognition using pca-ica and fuzzy logic classifier," *Journal of Computer Science*, vol. 3, no. 12, pp. 915–920, 2007.

[121] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[122] A. G. Howard, M. Zhu, B. Chen, D. Wang, W. Weyand, S. Andrychowicz, B. Chen, Z. Kalenichenko, V. Rusol, J. Shlens, and et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 650–658.

[123] J. Deng, J. Guo, J. Xue, and S. Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5203–5212.

[124] M. Tyagi, "Viola Jones Algorithm and Haar Cascade Classifier," Jul. 2021. [Online]. Available: https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8/

[125] "The Viola-Jones Algorithm | Baeldung on Computer Science," Oct. 2022. [Online]. Available: https://www.baeldung.com/cs/viola-jones-algorithm

[126] K. Mittal, "A Gentle Introduction Into The Histogram Of Oriented Gradients," Mar. 2024. [Online]. Available: https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa

[127] V. S. Mishra, "An Efficient Face Detection System using the Viola-Jones Algorithm."

[128] R. Chaudhury, "Adaboost classifier for face detection using viola jones algorithm," Jan. 2022. [Online]. Available: https://medium.com/@rohan.chaudhury.rc/adaboost-classifier-for-face-detection-using-viola-jones-algorithm-30246527db11

[129] R. Academy and C. L. Platform, "Recforge Academy - Gamified Online Learning," Aug. 2025, section: Education. [Online]. Available: https://academy.recforge.com/#/course/face-recognition-132/level-4-feature-extraction-methods/lbp-local-binary-patterns-for-texture-analysis

[130] L. S. López, "Local Binary Patterns applied to Face Detection and Recognition."

[131] B. Ahamed, "Face Recognition with Local Binary Patterns," Nov. 2023. [Online]. Available: https://medium.com/@sbasil.ahamed/face-recognition-with-local-binary-patterns-42e307f3f682

[132] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893. [Online]. Available: http://ieeexplore.ieee.org/document/1467360/

[133] "HOG Feature Descriptor-An Intuitive Overview," Jul. 2023. [Online]. Available: https://subham73.github.io/posts/HOG-Feature-Descriptor/

[134] s. jack, "Face detection using dlib HOG," Jul. 2020. [Online]. Available: https://medium.com/mlcrunch/face-detection-using-dlib-hog-198414837945

[135] S. R. Rath, "Face Detection with Dlib using HOG and Linear SVM," Jun. 2021. [Online]. Available: https://debuggercafe.com/face-detection-with-dlib-using-hog-and-linear-svm/

[136] A. Mittal, "Haar Cascades, Explained," Jan. 2024. [Online]. Available: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

[137] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," Apr. 2019, arXiv:1807.05511 [cs]. [Online]. Available: http://arxiv.org/abs/1807.05511

[138] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection."

[139] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[140] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016, p. 21–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[141] S. Rath, "OpenCV DNN Module and Deep Learning (A Definitive guide)," Apr. 2021. [Online]. Available: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/

[142] I. Kontaev, "Face Recognition with ArcFace — learnopencv.com," https://learnopencv.com/face-recognition-with-arcface/, [Accessed 20-08-2025].

[143] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[144] "OpenCV: DNN-based Face Detection And Recognition." [Online]. Available: https://docs.opencv.org/4.x/d0/dd4/tutorial_dnn_face.html?utm_source=chatgpt.com

[145] G. Boesch, "MediaPipe: A Guide Google's Open Source Framework - viso.ai — viso.ai," https://viso.ai/computer-vision/mediapipe/, [Accessed 19-08-2025].

[146] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," 2019. [Online]. Available: https://arxiv.org/abs/1907.05047

[147] "What is MediaPipe? A Guide for Beginners — blog.roboflow.com," https://blog.roboflow.com/what-is-mediapipe/, [Accessed 19-08-2025].

[148] "GitHub - google-ai-edge/mediapipe: Cross-platform, customizable ML solutions for live and streaming media. — github.com," https://github.com/google-ai-edge/mediapipe, [Accessed 19-08-2025].

[149] "YOLOv8: State-of-the-Art Computer Vision Model — yolov8.com," https://yolov8.com/, [Accessed 20-08-2025].

[150] Ultralytics, "YOLOv8 — docs.ultralytics.com," https://docs.ultralytics.com/de/models/yolov8/, [Accessed 20-08-2025].

[151] "GitHub - ultralytics/ultralytics: Ultralytics YOLO — github.com," https://github.com/ultralytics/ultralytics, [Accessed 20-08-2025].

[152] "InsightFace — insightface.ai," https://insightface.ai/, [Accessed 20-08-2025].

[153] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 5962–5979, 2022.

[154] "GitHub - deepinsight/insightface: State-of-the-art 2D and 3D Face Analysis Project — github.com," https://github.com/deepinsight/insightface, [Accessed 20-08-2025].

[155] "SSD object detection explained: (SSD vs YOLO) — tredence.com," https://www.tredence.com/blog/from-ssd-to-yolo-introduction-to-modern-object-detection, [Accessed 20-08-2025].

[156] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[157] R. A. Nabid, S. Pranto, N. Mohammed, F. Sarker, M. Huda, and K. Mamun, *AI Reception: An Intelligent Bengali Receptionist System Integrating with Face, Speech, and Interaction Recognition*, 09 2022, pp. 76–91.

[158] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," 2018. [Online]. Available: https://arxiv.org/abs/1710.08092

[159] J. Wang, Y. Yuan, B. Li, G. Yu, and S. Jian, "Sface: An efficient network for face detection in large scale variations," 2018. [Online]. Available: https://arxiv.org/abs/1804.06559

[160] X. Li, H. Qiu, L. Wang, H. Zhang, C. Qi, L. Han, H. Xiong, and H. Li, "Challenges and trends in egocentric vision: A survey," version: 3. [Online]. Available: http://arxiv.org/abs/2503.15275

[161] Introduction to project aria docs | project aria docs. [Online]. Available: https://facebookresearch.github.io/projectaria_tools/docs/intro

[162] Meta's reality labs launches aria 2 AR research glasses. Section: article. [Online]. Available: https://displaydaily.com/metas-reality-labs-launches-aria-2-ar-research-glasses/

[163] Meta-AI glasses | ray-ban meta und oakley meta. [Online]. Available: https://www.meta.com/de/ai-glasses/

[164] Egocentric 4d perception (EGO4d). [Online]. Available: https://ego4d-data.org/

[165] EgoLife: Towards egocentric life assistant | AI research paper details. arXiv. Zugriff am Juli 9, 2025, https://arxiv.org/html/2503.03803v1. [Online]. Available: https://aimodels.fyi

[166] EgoLife: Towards egocentric life assistant | AI research paper details. arXiv. Zugriff am Juli 9, 2025, https://arxiv.org/html/2503.03803v1. [Online]. Available: https://aimodels.fyi

[167] "Egocentric 4D Perception (EGO4D) — ego4d-data.org," https://ego4d-data.org/, [Accessed 20-08-2025].

[168] X-LeBench: A benchmark for extremely long egocentric video understanding. [Online]. Available: https://arxiv.org/html/2501.06835v1

[169] Seeing in the dark: Benchmarking egocentric 3d vision with the oxford day-and-night dataset. [Online]. Available: https://arxiv.org/html/2506.04224v1

[170] How AI smart glasses are changing everyday experiences. Section: VERTU CREATIVE. [Online]. Available: https://vertu.com/ai-tools/ai-smart-glasses-transform-daily-life/

[171] Vinci: A real-time embodied smart assistant based on egocentric vision-language model. [Online]. Available: https://arxiv.org/html/2412.21080v1

[172] Vinci: A real-time embodied smart assistant based on egocentric vision-language model. [Online]. Available: https://arxiv.org/html/2412.21080v1

[173] Eye tracking applications for AI smart glasses | SeeTrue technologies. [Online]. Available: https://www.seetruetechnologies.com/ai-powered-smart-glasses-with-eye-tracking-unlocking-context-aware-intelligence/

[174] C. M. Fang, Y. Samaradivakara, P. Maes, and S. Nanayakkara, "Mirai: A wearable proactive AI ïnner-voice"for contextual nudging," version: 1. [Online]. Available: http://arxiv.org/abs/2502.02370

[175] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. [Online]. Available: http://ieeexplore.ieee.org/document/6619194/

[176] A. Rathore, P. Nagar, C. Arora, and C. Jawahar, "Generating 1 minute summaries of day long egocentric videos," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, pp. 2305–2313. [Online]. Available: https://dl.acm.org/doi/10.1145/3343031.3350880

[177] D. Wen, "Comparing learning paradigms for egocentric video summarization," version: 1. [Online]. Available: http://arxiv.org/abs/2506.21785

[178] Video summarization techniques: A comprehensive review. [Online]. Available: https://arxiv.org/html/2410.04449v1

[179] J. Xut, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singht, "Gaze-enabled egocentric video summarization via constrained submodular maximization," vol. 2015, pp. 2235–2244. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4784707/

[180] admin. Addressing the challenge of partial occlusion detection in facial recognition system. [Online]. Available: https://facia.ai/blog/addressing-the-challenge-of-partial-occlusion-detection-in-facial-recognition-system/

[181] A. K. Singh, A. Singh, H. Sirohi, and N. Bhardwaj, "Review of challenges and innovations in occluded facial recognition in disguise and crowd," vol. 12, no. 5, pp. 1481–1487, publisher: International Journal for Research in Applied Science and Engineering Technology (IJRASET). [Online]. Available: https://www.ijraset.com/best-journal/review-of-challenges-and-innovations-in-occluded-facial-recognition-in-disguise-and-crowd

[182] Y. Huang, Z. Wu, J. Chen, and H. Xiang, "Privacy-preserving face recognition method based on randomization and local feature learning," vol. 10, no. 3, p. 59, publisher: MDPI AG. [Online]. Available: https://www.mdpi.com/2313-433X/10/3/59

[183] How voice assistants revolutionize human-computer interaction | aiello. Section: AVA. [Online]. Available: https://aiello.ai/blog-en/voice-assistants-human-computer-interaction/

[184] Key features and uses of AI voice assistants. [Online]. Available: https://www.ema.co/additional-blogs/addition-blogs/key-features-and-uses-of-ai-voice-assistants

[185] "Breaking the silence: Innovation in wake word activation," *Malque Publishing*, 2025, zugriff am Juli 9, 2025, https://www.malque.pub/ojs/index.php/msj/article/download/899/584.

[186] Top 4 speech recognition challenges & solutions in 2025. [Online]. Available: https://research.aimultiple.com/speech-recognition-challenges/

[187] Y. Chen, H. Li, S.-Y. Teng, S. Nagels, Z. Li, P. Lopes, B. Y. Zhao, and H. Zheng, "Wearable microphone jamming," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3313831.3376304

[188] W. Zhou, K. Cao, H. Zheng, X. Zheng, M. Liu, P. O. Kristensson, W. Mayol-Cuevas, F. Zhang, W. Lin, and J. Shen, "X-LeBench: A benchmark for extremely long egocentric video understanding." [Online]. Available: http://arxiv.org/abs/2501.06835

[189] P. Nagar, A. Rathore, C. V. Jawahar, and C. Arora, "Generating personalized summaries of day long egocentric videos," vol. 45, no. 6, pp. 6832–6845, publisher: Institute of Electrical and Electronics Engineers (IEEE). [Online]. Available: https://ieeexplore.ieee.org/document/9562265/

[190] Smart glasses & facial recognition: Privacy concerns & future. [Online]. Available: https://www.datainsightsmarket.com/news/article/smart-glasses-facial-recognition-privacy-concerns-future-49367

[191] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[192] "Labelled Faces in the Wild (LFW) Dataset — kaggle.com," https://www.kaggle.com/datasets/jessicali9530/lfw-dataset, [Accessed 23-08-2025].

[193] S. B. Ahmad, S. Ali, J. Ahmad, M. Adnan, and M. Fraz, "On the frontiers of pose invariant face recognition: a review," *Artificial Intelligence Review*, vol. 53, 04 2020.

[194] "openslr.org — openslr.org," https://www.openslr.org/12, [Accessed 23-08-2025].

[195] "The LJ Speech Dataset — keithito.com," https://keithito.com/LJ-Speech-Dataset/, [Accessed 23-08-2025].

[196] Anonymous, "Multilingual performance analysis of large language models," in *Submitted to ACL Rolling Review - April 2024*, 2024, under review. [Online]. Available: https://openreview.net/forum?id=tkbIJpb6tO

[197] I. Bigoulaeva, H. T. Madabushi, and I. Gurevych, "The inherent limits of pretrained llms: The unexpected convergence of instruction tuning and in-context learning capabilities," 2025. [Online]. Available: https://arxiv.org/abs/2501.08716

[198] "- YouTube — youtube.com," https://www.youtube.com/shorts/HiohoQccNhg, [Accessed 23-08-2025].

Eigenständigkeitserklärung

Alle in diesem Dokument zitierten Sätze oder Textstellen aus den Werken anderer Personen wurden durch eindeutige Querverweise auf Autor, Werk und Seite(n) ausdrücklich kenntlich gemacht. Alle Abbildungen, die nicht vom Verfasser dieses Berichts stammen, wurden mit der ausdrücklichen Genehmigung des Urhebers verwendet und sind entsprechend gekennzeichnet. Ich bin mir bewusst, dass ein Versäumnis in dieser Hinsicht als Plagiat gilt und als Grund für ein Nichtbestehen angesehen wird.

Name: Kerem Deniz Yasar

Unterschrift: 

Datum: 09.01.2026