Müdigkeitsdetektor

Projektarbeit

vorgelegt von
Jan-Nicolas Weider, Felix Tischler und Mattis Dietrich

angefertigt am
Lehrstuhl für Digitale Bildverarbeitung
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena

Projekt: Projekt Intelligente Systeme
Betreuer: Niklas Penzel
Datum: 30. September 2023

Inhaltsverzeichnis

1 Einführung

Die Anwendung von digitaler Bildverarbeitung und maschinellem Lernen hat in den letzten Jahren eine transformative Bedeutung erlangt und spielt eine zunehmend entscheidende Rolle in einem breiten Kontingent von verschiedenen Endnutzern. Diese Technologien ermöglichen es, visuelle Informationen präzise zu analysieren, Muster zu erkennen und darauf basierend intelligente Entscheidungen zu treffen. Die Relevanz dieser Technologien erstreckt sich über eine Vielzahl von Anwendungsgebieten, von der Medizin über die Robotik bis hin zur Automobilindustrie und zur Sicherheitstechnik.

1.1 Aufgabenstellung

Im Rahmen der Bachelorstudiengänge Informatik und Angewandte Informatik an der Friedrich-Schiller-Universität Jena wird die Möglichkeit geboten, das Modul „Projekt Intelligente Systeme“ zu belegen. Da in dem Bereich Rechnersehen und Mustererkennung bereits erste Kenntnisse vermittelt wurden und sich ein Interesse ausbilden konnte, ist es naheliegend, die erlangten theoretischen Fähigkeiten mit einem Projekt in die Praxis umzusetzen. Infolgedessen entschied man sich für dieses Modul, um in Gruppenarbeit aktuelle Techniken und Methodiken der digitalen Bildverarbeitung und des maschinellen Lernens weiter zu erproben. Dabei sollten bestehende Verfahren nach Stand der Kunst genutzt und sinnvoll eingesetzt werden, um ein umfassenderes Bildverarbeitungsprogramm zu entwickeln. Der Umfang dieses Projektes reicht von der Projektplanung und Konzeptionierung über die Dokumentation bis zur Vorstellung der fertigen Anwendung und der Anfertigung eines Abschlussberichts. Dabei soll man sich zu Beginn mit der Formulierung der Projektzielen nach der SMART-Methode auseinandersetzen. Anschließend sind Arbeitspakete und erste Zuständigkeiten auszuformulieren, welche in einem Gantt-Diagramm zur Planung wiederzufinden sind. Das Projekt wird in Gruppenarbeit angefertigt und von Meilenstein-Meetings zur Präsentation des aktuellen Standes bis zur Vorstellung dem Arbeitsplan gemäß entwickelt. Welche Thematik die Anwendung gezielt anspricht, ist den Gruppenmitgliedern freigestellt und unterliegt deren Interesse und Motivation.

1.2 Motivation

Durch die thematische Freistellung musste zu Beginn festgelegt werden, welcher Anwendungsbereich von Interesse ist. Schnell herrschte Einigkeit darüber, dass man etwas

entwickeln möchte, was eine große Menge potenzieller Anwender anspricht. Das Projekt sollte demzufolge ein Ergebnis liefern, welches im Alltag anwendbar ist und bestenfalls zusätzlich einen gesellschaftlichen Nutzen aufweist, in dem für bestimmte Vorgehen die Sicherheit erhöht wird. Eine Erhöhung der Sicherheit ist im stressgeprägten Alltag, wobei Vieles schnellstmöglich vonstattengehen soll, von großer Relevanz. Vor allem im Straßenverkehr herrscht durch die hohe Grundgeschwindigkeit ein erhöhtes Risiko, weshalb kurze Unaufmerksamkeiten, durch beispielsweise Sekundenschlaf, bereits schwerwiegende Folgen mit sich ziehen können. In diesem Bereich kann die Kontrolle des Aktivitätsstatus von großer Bedeutung sein, da hiermit die Sicherheit aller Verkehrsteilnehmer positiv beeinflusst werden kann.

Weitere Anwendungen eines solchen Müdigkeitsdetektors können in der aktiven Kontrolle ausgewählter Mitarbeiter mit sicherheitsrelevanten Jobs oder auch in Online-Klassen zur Aktivitätsüberwachung der Teilnehmer sein. All diese Anwendungsgebiete, welche einen bleibenden Einfluss auf die Gesellschaft hinterlassen können, motivieren uns dazu, einen Müdigkeitsdetektor im Rahmen dieses Projekts zu entwickeln. Das Programm soll über eine Webcam laufen, die Person wird detektiert und verschiedene signifikante Merkmale für Müdigkeitsanzeichen werden stetig überprüft. Weiterhin sollen potenzielle Ausfälle frühzeitig erkannt und durch eine Warnung verhindert werden, somit kann ein wesentlicher Teil zur Sicherheit beigetragen werden.

1.3 Arbeitspakete und Gantt-Diagramm

Aus der gewählten Thematik wurden somit die Arbeitspakete abgeleitet, welche durchlaufen werden mussten, um das Projekt zum Abschluss bringen zu können. Diese gemeinsamen auszuarbeiten und zu besprechen stellte den Start des Moduls dar. Hierbei wurden abzuarbeitende Arbeitspakete von der Einrichtung der Entwicklungsumgebung über Implementationen diverser Funktionalitäten bis zur Optimierung und schlussendlich Präsentation der Arbeit formuliert. Um den zeitlichen Ablauf planen zu können wurde ein Gantt-Diagramm angelegt, welches sowohl die zeitliche Einteilung als auch die Zuständigkeiten über ausgewählte Arbeitspakete festlegt. Durch das Erstellen dieser Übersicht sollte die Fähigkeit der eigenen Projektplanung erlernt werden. An diesem Entwurf wurde sich orientiert und erste Umsetzungen starteten.

2 Theorie

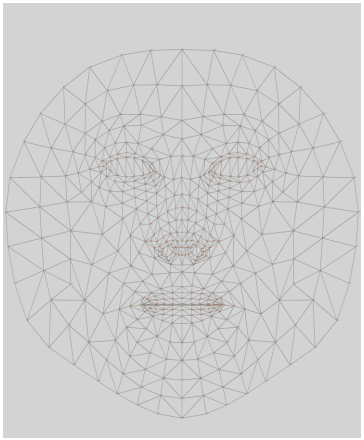
Zur Müdigkeitserkennung im Zusammenhang mit dem Autofahren gibt es verschiedene Methoden, wie sie von Arunasalam et al. [AYA+20] und Ramzan et al. [RKA+19] beschrieben werden. Diese lassen sich in drei Hauptkategorien unterteilen:

- Physiologische Methoden: Hierzu gehören Messungen wie Puls und EEG.
- Fahrverhaltensbasierte Methoden: Dies umfasst die Analyse des Fahrverhaltens mittles Sensoren am Auto.
- Verhaltensbasierte Methoden: Hierbei liegt der Fokus auf Beobachtungen von Auge, Gesicht und Kopfbewegungen.

Die visuelle Analyse des Verhaltens bietet eine praktikable Lösung, da diese auch vielfältig einsetzbar ist, bepsilesweise auch um Müdugkeit beim Lernen und Arbeiten zu erkennen.

Unser Hauptziel, wie bereits in der Aufgabenstellung erwähnt, besteht darin, eine frühzeitige Warnung auszugeben, wenn eine Person Anzeichen von Müdigkeit zeigt. Wir möchten damit eine Methode entwickeln, um Sekundenschlaf frühzeitig zu verhindern. Dies erfordert die Erkennung von Müdigkeit sowie die Identifizierung von Ablenkung (z. B. wenn eine Person zu lange nicht nach vorne schaut). Die Herausforderung besteht darin, den Zeitpunkt zu bestimmen, ab dem eine Person als müde gelten kann, und objektive Kriterien dafür zu entwickeln. Eine weitere wichtige Herausforderung besteht darin, dass Müdigkeit bei jeder Person unterschiedlich ist und sich visuelle Verhaltensweisen individuell manifestieren, beispielsweise variiert die Blinzelrate erheblich von Person zu Person, selbst bei Ruhe (Bentivoglio et al. [BBC+97] berichtet von 4 bis zu 48 Blinzelschlägen pro Minute).

Unsere Vorgehensweise orientiert sich an Ghodoosian et al. [GGA] und verwendet deren Datensatz als Ausgangspunkt. Wir verarbeiten Videodaten einer Person und extrahieren Landmarks, um quantitative Merkmale zu berechnen. Unsere Schwerpunkte liegen auf den visuellen Eigenschaften des Auges, wobei die Erkennung von Blinzeln eine entscheidende Rolle spielt. Um die Ergebnisse auf individuelle Unterschiede abzustimmen, führen wir eine Kalibrierung durch, die absolute Daten in relative Werte umwandelt. Diese Merkmale werden dann für die Klassifikation von „müde“ und „nicht müde“ verwendet, woraufhin bei „müde“ immer wieder eine Warnung ertönt. Eine akustische Warnung ertönt ebenfalls, wenn eine Person in Sekundenschlaf verfällt und die Augen über einen längeren Zeitraum geschlossen sind.



(a) Die Landmarks des Mediapipe Face Mesh mit Nummerierungen.

(b) Alle Landmarks des Face Mesh während der Anwendung.

(c) Die Augenlandmarks des Face Mesh zur Berechnung der EAR.

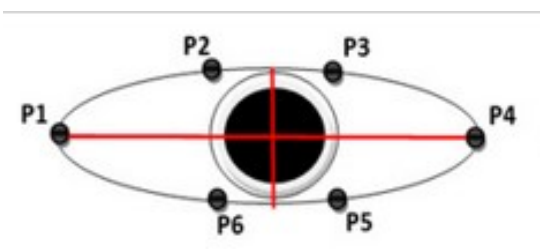
Abbildung 1: Prozess der Extraktion der entscheidenden Landmarks für unsere Müdigkeitsdetektion.

2.1 Gesichtserkennung

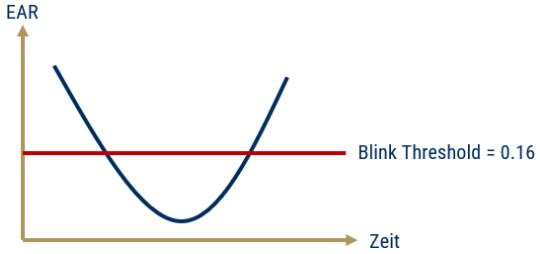
Um verwertbare Daten zu generieren, ist die Erkennung der Augenstruktur in Echtzeit entscheidend, um Blinezlschläge zu identifizieren, was die Grundlage für weitere Analysen bildet. Eine höhere Bildwiederholrate (auch Framerate; Bilder pro Sekunde) ermöglicht präzisere Ergebnisse, obwohl die meisten Kameras und Videodaten üblicherweise eine Framerate von etwa 30 Bildern pro Sekunde aufweisen. Es ist von Bedeutung, dass die Framerate durch die Berechnungen des Algorithmus nicht erheblich verlangsamt wird.

Ein leistungsstarkes und schnelles System dafür ist Mediapipe [LTN+]. Das Mediapipe Face Mesh erkennt in Echtzeit 468 3D-Gesichtslandmarks, selbst auf mobilen Geräten, und ermöglicht die Approximation einer 3D-Oberfläche des Gesichts durch maschinelles Lernen. Für unsere Anwendung verwenden wir Version 0.9.0, um eine Integration mit der Kivy-Bibliothek zu ermöglichen, da eine neuere Version von Mediapipe auf unerklärliche Probleme stößt. In der Version 0.9.0 des Mediapipe Face Mesh bleibt die Grundeinstellung statisch und bewegt sich als Ganzes mit, was jedoch für unsere Untersuchung unzureichend ist. Daher muss die Einstellung „refine“ auf „true“ gesetzt werden, um zehn zusätzliche Landmarks und Bewegungen innerhalb des Gesichts zu erfassen. Darüber hinaus muss der „static image mode“ auf „false“ gesetzt werden, um das Face Mesh auf Videos anzuwenden.

Wir haben auch andere Ansätze für ein Face Mesh evaluiert, darunter den Dlib Face



(a) Die sechs Landmarks von P1 bis zu P6 in ihrer Anordnung am Auge [DCC+22]



(b) Vereinfachte Darstellung eines Blinzelvorgangs und der Auswirkung auf die EAR.

Abbildung 2: Prozess der Berechnung eines Blinzlers über die sechs Augenlandmarks (a) über die EAR Kurve und dem Blink Threshold (b)

Landmark Detector (Davis E. King [Kin09]), basierend auf Dalal et al. [DT05]. Im Vergleich zum Mediapipe Face Mesh erzielte dieser jedoch in Bezug auf Genauigkeit schlechtere Ergebnisse und zeigte gelegentlich Aussetzer. Weitere Details dazu befinden sich in der Evaluation (siehe Kapitel 3.2).

2.2 Blinzeldetektion

Durch die Berechnung von Landmarks können wir nun das Blinzeln einer Person erkennen. Eine verbreitete Methode hierfür ist die Berechnung der „Eye Aspect Ratio“ (EAR), wie von Soukupova [Sou16] präsentiert. Die EAR wird anhand des Auges und sechs Landmarks (P1 bis P6) ermittelt und entspricht dem Verhältnis der vertikalen Öffnung zur horizontalen Öffnung:

$$\text{EAR} = \frac{\|P2 - P6\| + \|P3 - P5\|}{2\|P1 - P4\|} \tag{1}$$

In dieser Formel repräsentieren P1 bis P6 die Koordinaten der Landmarks, wobei P1 und P4 die äußeren Augenwinkel sind, P2 und P6 die oberen Augenlider und P3 und P5 die unteren Augenlider (Abbildung 2a). Die $\|\|$ Symbole stellen die euklidische Norm dar, die den Abstand zwischen den Punkten berechnet.

Die EAR ist stets größer, wenn das Auge geöffnet ist, im Vergleich dazu, wenn es geschlossen ist. Allerdings variieren diese Werte zwischen Personen, wie auch unser eigener Test zeigt, beispielsweise beträgt der durchschnittliche Wert für Proband 1 circa 0.295, wohingegen Proband 2 einen Wert von 0.252 aufweist (siehe Kapitel 3.2). Der Durchschnitts-EAR wird über beide Augen berechnet und sichert die kontinuierliche Erfassung dieses Werts für jedes Einzelbild, was weitere Analysen ermöglicht:

$$\text{Durchschnittliche EAR} = \frac{\text{EAR}_{\text{linkes Auge}} + \text{EAR}_{\text{rechtes Auge}}}{2} \tag{2}$$

Bei einem Blinzelschlag zeigt die EAR-Kurve einen typischen Verlauf, der in Abbildung 2b dargestellt ist - sie sinkt zunächst parabelförmig ab und steigt dann wieder an. Ein wichtiger Aspekt ist nun der Schwellenwert, um zu erkennen, ab wann das Auge als geschlossen oder offen gilt, um ein Blinzeln zu detektieren und seine Dauer zu berechnen. In unseren Selbsttests (siehe Kapitel 3.2) hat sich vorläufig ein fester Schwellenwert von 0.16 als am besten geeignet erwiesen. Sobald die EAR den Schwellenwert unterschreitet, wird das Auge als geschlossen betrachtet, und wenn er wieder überschritten wird, wird der Vorgang als ein vollständiges Blinzeln erkannt. Die Blinzelzeit kann anhand der Anzahl der Frames ermittelt werden, in denen der EAR-Wert kleiner als 0.16 ist. Um verschiedene Bildwiederholungsraten auszugleichen, wird die Framerate des Videos verwendet, um einen zeitlichen Wert in Millisekunden zu erhalten. Dies ist entscheidend für die Trainingsphase des Klassifikators und die Anwendbarkeit über verschiedene Bildwiederholungsraten und Geräte hinweg. Allerdings sollten weitere Untersuchungen zur Verbesserung der Genauigkeit sowie zur Berücksichtigung individueller Unterschiede und möglicher Augenkrankheiten wie Grauer Star oder altersbedingter Verengung der Augen in zukünftigen Arbeiten durchgeführt werden. Eine agile Berechnung der EAR-Schwelle zu Beginn jeder Aufnahme für jede Person könnte eine Möglichkeit sein, wie es von Ghoddoosian et al. [GGA] in ihrem „Blink Retrieval Algorithm“ beschrieben wird.

2.3 Features

Der nächste Schritt in unserem Verarbeitungsprozess besteht darin, Features zu sammeln. Insbesondere für das Auge stehen verschiedene mögliche Features zur Verfügung und unterschiedliche Kombinationen können zu besseren Ergebnissen führen, wie von Dreißig et al. [Mar] untersucht wurde. Dreißig et al. hat zusätzlich Kopfbewegungen in seine Untersuchungen einbezogen. Auch Ebrahim [Par16] stellt 19 Features vor, die sich ausschließlich auf den Blinzelvorgang beziehen.

Für uns haben sich folgende Features bewährt: die einfache Betrachtung der „Eye Aspect Ratio“ (EAR), die durchschnittliche Blinzeldauer und der „Percentage of Eye Closure“-Wert (PERCLOS). Die EAR bildet die Grundlage für weitere Features und besitzt auch allein schon eine gewisse Aussagekraft. Wir betrachten die gemittelte EAR über eine Zeitspanne und vergleichen sie mit vorherigen Zeitspannen. Außerdem analysieren wir die gemittelte EAR über alle Werte kleiner als 0.16, um die Werte zu entfernen,

wenn das Auge als geschlossen gilt.

Ein weiteres Feature ist die durchschnittliche Blinzeldauer. Wenn eine Person müde wird, sollte die Blinzeldauer länger sein. Daher speichern wir für jeden Blinzelvorgang die Dauer in Millisekunden und berechnen den Durchschnitt über eine Zeitspanne. Der PERCLOS-Wert, erstmals von Wierwille et al. [WWK+94] beschrieben, repräsentiert den Anteil der Zeit in einer Minute, in der die Augen zu mindestens 80 Prozent geschlossen sind. Bei müden Personen wird dieser Wert tendenziell größer. Für eine einfachere Berechnung verwenden wir den PERCLOS-Wert als den Anteil der Zeit in einer Minute, in der der EAR-Wert unterhalb des Schwellenwerts von 0.16 liegt, was darauf hinweist, dass das Auge als geschlossen betrachtet wird.

Die Frage nach der Festlegung der Zeitspanne für diese Berechnungen stellt sich nun als nächstes. Unsere Recherchen ergaben, dass eine Dauer von 60 Sekunden als angemessene Spanne angesehen wird, da Werte wie PERCLOS pro Minute berechnet werden. Diese Berechnungen erfolgen überlappend, was bedeutet, dass bei jedem neuen Frame neue Werte für die Features generiert werden.

Ein Beispielhafter Feature-Vektor nach der Berechnung ist in Tabelle 1 zu erkennen.

	PERCLOS (%)	**Blinzeldauer (ms)**	**EAR bei geöffnete Augen**	**EAR**
Wach	0.0094	80.24	0.2639	0.2627
Fraglich	0.0879	174.77	0.2267	0.2188
Müde	0.1780	168.60	0.1985	0.1878

Tabelle 1: Beispielhafter Featurevektor

2.4 Kalibrierung

Eine spezifische Herausforderung, die wir in Anbetracht unserer Aufgabenstellung betrachten müssen, ist die Individualität in Bezug auf die Augenöffnung und das Verhalten bei Müdigkeit. Bei verschiedenen Personen zeigt sich Müdigkeit auf unterschiedliche Weisen, bei einer Person kann die Blinzeldauer länger werden, während bei einer anderen Person eher die Blinzelhäufigkeit zunimmt. Zusätzlich variiert die EAR bei jeder Person erheblich, wie auch unser Selbsttest (siehe Kapitel 3.2) bestätigt hat. Daher ist eine Relativierung der Werte unerlässlich.

Zu Beginn jeder Aufzeichnung gehen wir davon aus, dass die Person wach ist und verwenden die ersten 60 Sekunden, um Referenzwerte für den Wachzustand zu erfassen. Diese Zeitspanne entspricht derjenigen, die für die weiteren Berechnungen verwendet wird. Die für jeden neuen Frame berechneten Werte werden in Beziehung zu den Werten

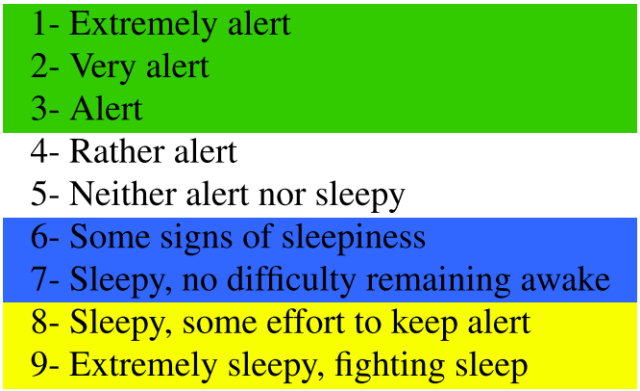


Abbildung 3: Karolinska Sleepiness Scale (KSS)

des Wachzustands gesetzt. Auf diese Weise erhalten wir immer relative Werte für jedes einzelne Feature.

2.5 Datensatz

Bei dem ausgewählten Datensatz zum Trainieren des Klassifikators handelt es sich um den UTA Real-Life Drowsiness Dataset von Ghodoosian et al. [GGA]. Der Datensatz stand unglücklicherweise nicht vollständig zur Verfügung, da auf der offiziellen Seite der Veröffentlichenden technische Probleme zur Zeit der Ausarbeitung vorlagen. In dem verwendeten Datensatz befinden sich jeweils drei Videos von 48 Individuen. Die teilnehmenden Personen wurden so ausgewählt, dass verschiedene Ethnien, Altersgruppen und Geschlechter abgedeckt sind. Weiterhin trug ein Teil der Probanden eine Brille. Die Aufnahmen entstanden jeweils in natürlicher Umgebung mit verschiedenen Perspektiven, Hintergründen und Belichtungsverhältnissen. Pro Person ist ein Video für die Klasse „wach“, „fraglich“ und „müde“ gegeben.

Die drei Kategorien ergeben sich, wie in Abbildung 3 erkenntlich, durch eine Zusammenfassung des offiziellen Karolinska Sleepiness Scale (KSS). Die KSS wurde von Akerstedt et al. [AG90] zuerst vorgestellt. Als „wach“ werden die Stufen eins bis drei betitelt, wobei für „fraglich“ die Stufen sechs und sieben repräsentativ sind. Auf den Zustand „müde“ weisen die Einordnungen in die Stufen acht und neun hin. Die durchschnittliche Länge eines einzelnen Videos beträgt in etwa zehn Minuten, so dass die für die Klassifikation benötigten Featurewerte über einen längeren Zeitraum möglichst genau und fehlerunabhängig ermittelt werden können.

Zu beachten bei der vorliegenden Datengrundlage ist die Tatsache, dass die Videos über verschiedene Dateiformate, Frameraten und Längen verfügen. Einerseits sichert

dies die Robustheit des Programms, um mit verschiedener Hardware umgehen zu können, jedoch war andererseits für das automatisierte Einlesen und Featurextrahieren ein Mehraufwand benötigt.

Es wurde ein Skript erstellt, welches alle Videos mit dem entsprechenden Datenformat aufruft und abspielt und dabei die Extraktion der Features über eine bestimmte Zeit laufen lässt, sodass aus jedem Video relevante Informationen gewonnen werden, alle Werte aus den Videos gleich stark gewichtet sind und zusätzlich kein Video vor einer vollständigen Extraktion aufgrund zu kurzer Aufnahmedauer eines Probanden abbricht.

2.6 Klassifikation

In Bezug auf die drei Videos pro Person setzen wir für jedes Feature den Zustand „fraglich“ und „müde“ in Relation zum Wachzustand und führen daraufhin Klassifikationen basierend auf den relativen Werten durch. Hierbei konzentrieren wir uns auf drei häufig genutzte Klassifikatoren und nutzen die Funktionen aus der Bibliothek scikit-learn von Pedregosa et al. [PVG+11]. Die Ergebnisse werden anhand der folgenden drei Klassifikatoren ausgewertet und verglichen:

- Logistische Regression: Für zwei Klassen wird die „One-Versus-Rest“ (ovr) Multi-Class-Strategie verwendet, während die Strategie für drei Klassen automatisch ausgewählt wird. Das Modell nutzt den 'lbfgs'-Solver und führt bis zu 1000 Iterationen durch, um das Modell zu trainieren.
- K-Nearest Neighbour mit K=3.
- Support Vector Machine.

Zudem untersuchen wir die Ergebnisse für drei Klassen (müde/fraglich/wach) sowie für zwei Klassen (müde/wach). Wir zogen auch die Entwicklung eines Scores in Erwägung, haben diese Idee jedoch aufgrund der begrenzten Datenmenge verworfen, da uns lediglich Daten für die Zustände „müde“, „fraglich“ und „wach“ zur Verfügung standen, jedoch keine weiteren Zwischenwerte.

Die Details zu den Klassifikationsergebnissen werden in der Evaluation in Kapitel 3.3 genauer analysiert. Für unser Projekt hat sich jedoch der K-Nearest Neighbour als die beste Option erwiesen. Um das System weiter zu verbessern, wäre eine größere Datenmenge von entscheidender Bedeutung. Zusätzlich könnten auch weitere Klassifikatoren in Betracht gezogen werden.

2.7 Evaluationsmetriken zur Beurteilung der Klassifikatoren

Für die Evaluierung unserer Klassifikatoren haben wir verschiedene Metriken angewendet. Dabei haben wir die Funktionen aus der scikit-learn-Bibliothek von Pedregosa et al. [PVG+11] genutzt. Konkret haben wir den Precision Score, Recall Score, F1 Score, Accuracy Score und die Fehlermatrix (auch als Confusion Matrix bekannt) verwendet.

Der Precision Score gibt an, wie viele der vorhergesagten positiven Instanzen tatsächlich positiv sind. Sein Wert liegt zwischen 0 und 1, wobei 1 für eine perfekte Präzision steht.

Der Recall Score zeigt, wie viele der tatsächlich positiven Instanzen korrekt als positiv vorhergesagt wurden. Auch hier reicht der Wertebereich von 0 bis 1, wobei 1 für einen perfekten Recall steht.

Der F1 Score ist das harmonische Mittel zwischen Precision und Recall und bietet ein ausgewogenes Maß für die Modellleistung. Auch dieser Wert liegt zwischen 0 und 1, wobei 1 eine perfekte Balance zwischen Precision und Recall repräsentiert.

Der Accuracy Score gibt das Verhältnis der korrekt klassifizierten Instanzen zur Gesamtanzahl der Instanzen im Testdatensatz an, wobei höhere Werte auf eine bessere Klassifikationsgenauigkeit hinweisen.

Die Fehlermatrix, auch als Confusion Matrix bezeichnet, ist eine tabellarische Darstellung der Anzahl der wahren positiven, wahren negativen, falsch positiven und falsch negativen Vorhersagen eines Klassifikationsmodells. Sie bietet eine aufgeschlüsselte Übersicht über die Verteilung der Ergebnisse.

2.8 App-Entwicklung

Unser Projekt hat zum Ziel, eine Anwendung zu entwickeln, die die Müdigkeitserkennung durchführt. Diese App sollte auf verschiedenen Plattformen und Geräten zugänglich sein und es den Benutzern ermöglichen, ihre Müdigkeit in Echtzeit zu überwachen. Die App sollte einfach zu bedienen, effizient und zuverlässig sein.

2.8.1 Technologieauswahl

Zu Beginn der Technologieauswahl stand für uns bereits fest, dass Python das Mittel der Wahl für unsere Idee und Umsetzung aller benötigten Algorithmen ist. Wir begaben uns von Anfang an auf die Suche nach geeigneten Symbiosen, die uns die Entwicklung einer App mit Hilfe von Python ermöglichen.

Als folge dessen haben wir uns bei der Auswahl der weiteren Technologien für unser Projekt intensiv mit den verschiedenen Optionen auseinandergesetzt. Eine entscheidende Überlegung war die Notwendigkeit der Offline-Funktionalität, da unsere App in Umgebungen zum Einsatz kommen sollte, in denen eine dauerhafte Internetverbindung nicht immer gewährleistet ist. Aus diesem Grund haben wir uns bewusst gegen die Realisierung einer API mit React und Flask entschieden. Hinter der Entscheidung stand ein intensives Erproben und Evaluieren der Möglichkeiten mit React und Flask. Schlussendlich wurde jedoch deutlich, dass eine andere Lösung gefunden werden muss.

Nach genauerer Recherche, Erprobung und Evaluation fiel die Wahl auf Kivy. Die Entscheidung für das Kivy-Framework zur Frontend-Entwicklung war eine nahezu zwingende Konsequenz auf Grund von vielen Vorteilen. Kivy-Apps können auf verschiedenen Systemen zum Einsatz kommen und bieten somit eine breite Einsatzmöglichkeit. Zusätzlich synergiert Kivy mit Python und wurde einst für diese Programmiersprache entwickelt. Dies legte final den Grundstein für die Verbindung unseres Frontends mit unserer Logik, die ausschließlich auf Python basiert. Die Wahl von Kivy ermöglichte es uns, unser Projekt in den verschiedenen Fassetten starten und entwickeln zu können.

2.8.2 Entwicklungsprozess

Zu Beginn des Entwicklungsprozess war zwar entschieden, welche Technologien wir verwenden wollen, jedoch blieb die Frage der Implementierung noch aus. Es bildeten sich drei Teilbereiche, welche diese Frage beantworten konnten:

- Entwicklung des App-Grundgerüsts
- Entwicklung des Layouts der App
- Entwicklung der Integration von Logik und Layout

Ersteres definierte notwendige Schnittstellen und legte den Rahmen fest, in dem wir agierten. Es wurde eine Hauptdatei erstellt (mainmediapipe.py), welche die Logik des Backend aus einer externen Datei und ebenso das Layout beinhaltete und kombinierte. Zweiteres hingegen setzte alle notwendigen Entscheidungen bezüglich des Designs in einer Datei (mainmediapipe.kv) fest. Hierfür verwendeten wird das von Kivy mitgelieferte Dateiformat '.kv'. Letzteres beanspruchte am meisten Zeit, es mussten Verlinkungen zwischen der Logik und dem Layout vorgenommen werden. Es wurden Mechanismen für die visuelle und akustische Alarmierung entwickelt. Alles Schnittstellen wurden in Betrieb genommen und mussten untereinander kommunizieren können. Wir haben mittels der

uns zur Verfügung stehenden Möglichkeiten all diese Hürden bezwingen können und als Produkt des Entwicklungsprozesses ein funktionierendes Gesamtpaket entstehen lassen.

Zu diesem Zeitpunkt war die App rein desktopbasiert. Im Abschnitt 4.1 werden wir näher darauf eingehen, warum wir uns gegen die Fortsetzung der Bereitstellung bis zur Erstellung einer APK-Datei entschieden haben. Dabei werden wir auf die Herausforderungen und Lernprozesse eingehen, die diese Entscheidung beeinflusst haben.

2.8.3 Deployment-Strategie

Das große Ziel war, eine funktionierende Software zu haben, die unseren Projektanforderungen gerecht wird. Darüber hinaus hatten wir noch einen weiteren selbstgewählten Anspruch, wir wollten eine App entwickeln, die sich über eine maximale Verfügbarkeit auszeichnet. Das heißt konkret, das Ziel war sich sukzessive von Softwarecode, zu Desktop-App bis hin zur APK-Datei zu verbessern. Dies war ein Vorhaben, welches über die Anforderungen des Modul hinaus ging. Der Start der Deployment-Strategie stellte die Betrachtung folgender Faktoren dar:

- Zielplattformen und -geräte
- Deployment-Tools und -Methoden
- Herausforderungen bei der Bereitstellung

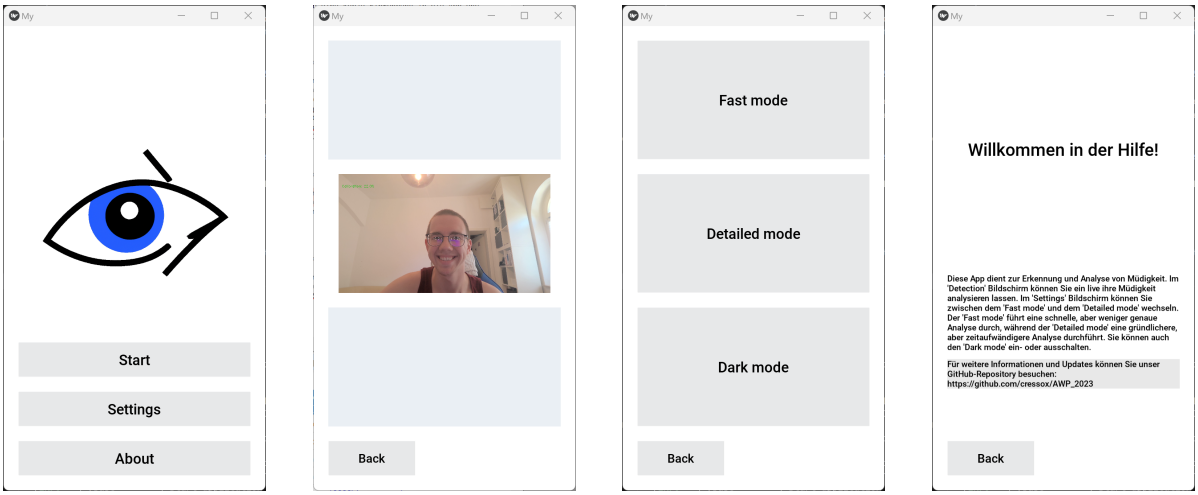
Zielplattformen und -geräte sollten zunächst Windows und Linux abbilden und in zweiter Instanz Android. Deployment-Tools und -Methoden waren und durch unsere Technologieauswahl bereits bindend vorgegeben. Herausforderungen bei der Bereitstellung ergaben sich dann zu häuft, denn die Möglichkeit, eine App möglichst unkompliziert bereit zu stellen, enttarnte sich als starke Fehleinschätzung. Wir konnten relativ unkompliziert unseren Code als Desktop-App anbieten, der auf Windows und Linux funktionierte, jedoch war der weg zur APK-Datei wesentlich widerstandsfähiger. Es folgten diverse Versuche, einen Build-Prozess anzustoßen, welcher unseren Code zur benötigten Datei umbauen sollte. Jedoch hatten wir einen entscheidenden Fehler in der Technologieauswahl begangen, wir legten uns von Anfang an auf Python fest und limitierten uns damit ausschlaggebend in der Wahl der Build-Möglichkeiten. Für eine nähere Betrachtung dieses Fehlers verweisen wir im Abschnitt 4.1.

3 Evaluation

Um die Genauigkeit des entwickelten Müdigkeitsdetektors einschätzen zu können, muss dieser evaluiert werden. Da verschiedene Verfahren und Metriken in die Abläufe involviert sind, ist nicht nur eine gesamtheitliche, sondern auch eine granulare Bewertung einzelner Stufen als sinnvoll anzusehen.

3.1 Das Interface

Die App bietet eine Benutzeroberfläche mit verschiedenen Bildschirmen, wie in der Abbildung 4 zu erkennen ist.



(a) Startseite (b) Detektion (c) Einstellungsseite (d) Hilfebereich

Abbildung 4: Verschiedene Screenshots aus der Anwendung.

In Abbildung 4a ist der Einstiegsbildschirm der App zu sehen. Hier erhalten Benutzer eine kurze Einführung in die App und können die Müdigkeitserkennung starten oder auf die Einstellungen zugreifen. Die eigentliche Müdigkeitserkennung findet im Erkennungsbildschirm (Abbildung 4b) statt, wo erkannte Gesichtsmerkmale wie die Augen markiert werden. Bei Erkennung von Müdigkeitsmerkmalen kann die App Warnungen anzeigen oder Alarmtöne abspielen. Im Einstellungsbildschirm (Abbildung 4c) können Benutzer verschiedene Einstellungen für die Benutzeroberfläche anpassen. Der Hilfebildschirm (Abbildung 4d) bietet detaillierte Informationen zur Verwendung der App und ihrer Funktionen.

Grundsätzlich sind wir mit dem Layout zufrieden, da es Minimalismus und Nutzbarkeit vereint. Der gesamte Funktionsumfang wird von der App abgedeckt und dem

Nutzer zur Verfügung gestellt. Hierbei ist die Bedienung selbsterklärend und zielführend, ohne die potenziellen Nutzer vor Schwierigkeiten der Orientierung zu stellen. Wie in Abschnitt 4.1 bereits erwähnt, hätte mehr Zeit in die Möglichkeiten der Entwicklung investiert werden können, somit hätte das Design noch interaktiver und flexibler sein können.

3.2 Selbsttest zu Metriken und Frameworks

Zur Evaluation verschiedener Schwellwerte zur Blinzeldetektion und Frameworks wurde einen Selbsttest im Wachzustand durchgeführt, wobei zu zweit in einer alltäglichen Arbeitsumgebung mit natürlicher Beleuchtung über die interne Webcam des eigenen Laptops das Programm circa eine Minute lang ausgeführt wurde. Von diesen Versuchen entnahmen wir die selbstgezählte reale und die detektierte Anzahl der Blinzelschläge in diesem Zeitraum für verschiedene Parametervoreinstellungen. Zusätzlich führten wir dies für zwei verschiedene Frameworks aus, um durch Selbsttests herauszufinden, welches für unseren Anwendungsbereich geeigneter ist. Für Proband 1 ist in Tabelle 2 zu erkennen, dass MediaPipe im Durchschnitt eine signifikant bessere Detektionsrate aufweisen kann als Dlib. Hierbei ist es wichtig, dass weder zu viele noch zu wenig Blinzelschläge detektiert werden. Da keine Toleranz für Fehldetektionen geplant war, ist dieser Wert zu optimieren. Für Probanden 2 sind ähnliche Ergebnisse zu verzeichnen, eine fehlerfreie Erkennung kann auch hier nur MediaPipe liefern. Jedoch ist zu bemerken, dass für diese Versuchsperson die optimale Detektion nur bei einem kleineren Schwellwert des EAR-Wertes gegeben ist. Zu begründen ist dies durch die generellen Unterschiede in Individuen sowie die persönliche Form und Anatomie des Auges. Für jede Person ist der EAR-Wert im geöffneten Zustand verschieden, ein kleinerer Wert tendiert dazu, dass bei einem vergleichsweise hohen Schwellwert einige Blinzelschläge fälschlicherweise erkannt werden, da der Schwellwert bereits ohne ein Schließen des Auges unterschritten wird. Dies ist auch in dem Test aufgefallen, so wurden für einige Kombinationen zu viele Blinzelschläge detektiert, weshalb der Detektor nicht vertrauenswürdig wäre. Ein zu kleiner Schwellwert resultiert jedoch darin, dass aufgrund der möglicherweise geringen Framerate der Videos ein Blinzeln nicht erkannt wird, wenn nicht zufälligerweise ein Bild bei der vollständigen Schließung des Auges aufgenommen wird. Nach Beachtung dieser Zusammenhänge und Auswertung der Tabelle des Selbsttest (Tabelle 2) ist der Entschluss zu ziehen, dass die Benutzung von MediaPipe als Framework das geeignetere Mittel ist. Weiterhin stellte sich der Schwellwert für die Blinzeldetektion von 0.16 als passend heraus, da dieser verschiedene Probleme minimiert. Für diesen Wert wurde die

höchste Genauigkeit erreicht, da sowohl die Anzahl der falschpositiven als auch falschnegativen Werte für das ausgewählte Framework gleich Null ist. Somit ist zu sagen, dass die implementierte Detektion verlässlich und genau arbeitet.

	Detektierte/reale Blinzelschläge			
	Proband 1		**Proband 2**	
Blink Threshold	**Dlib**	**MediaPipe**	**Dlib**	**MediaPipe**
0.16	33/43	45/45	54/53	54/54
	39/53	58/58	49/47	61/61
0.18	38/48	60/60	49/47	61/50
	33/52	60/53	57/52	42/41
0.20	50/52	64/64	65/44	60/37
	47/53	57/57	57/49	70/45
∅ **EAR**	0.32	0.295	0.259	0.252
∅ **PERCLOS**	0.08	0.063	0.109	0.107

Tabelle 2: Selbsttest zu verschiedenen Schwellwerten und Frameworks

Neben der Anzahl der Blinzelschläge wurden zusätzlich die durchschnittlichen EAR- und PERCLOS-Werte betrachtet, um eventuelle Zusammenhänge nachvollziehen zu können. Hierbei ist deutlich zu entnehmen, dass der EAR-Wert für Proband 2 deutlich kleiner und der PERCLOS-Wert deutlich größer als die des Probanden 1 sind. Dies spiegelt genau die Zusammenhänge der Metriken und der verschiedenen geeigneten Schwellwerts wider, da für Proband 2 ein kleinerer Schwellwert besser geeignet ist, was auf ein generell schwächer geöffnetes Auge hinweist. Dies würde in einem geringeren EAR-Wert resultieren, was die Daten verifizieren. Weiterhin sollte der PERCLOS-Wert höher sein, je größer die Anzahl an Blinzelschlägen oder je länger die Blinzeldauer pro Zeitspanne ist, da der Anteil, in dem die Augen geschlossen sind, größer ist. Unsere Beobachtungen stimmen mit diesem Verhalten überein, für Proband 1 wurde ein kleinerer Wert als für Proband 2 ermittelt.

3.3 Vergleich der Klassifikatoren

Wie in der Theorie im Abschnitt 2.6 zur Klassifikation beschrieben, wurden verschiedene Klassifikatoren getestet, um den am besten geeigneten für unsere Anwendung zu ermitteln. Dabei haben wir sowohl den Fall mit zwei Klassen (müde/wach) als auch den mit drei Klassen (müde/fraglich/wach) getestet. Die Metriken, die wir verwendet haben, wurden in Abschnitt 2.7 vorgestellt.

Die Trainingsdaten für die Klassifikatoren bestehen aus den berechneten Feature-Vektoren, die aus den Videos des verwendeten Datensatzes gewonnen wurden. Insgesamt

Klassifikator	F1-Score	Recall	Precision	Accuracy
KNN Klassifikator mit k = 3	0.743962	0.758621	0.801724	0.680542
Logistische Regression	0.663643	0.689655	0.671121	0.602709
Support Vector Machine	0.552622	0.62069	0.79454	0.489655

Tabelle 3: Ergebnisse der drei verglichenen Klassifikatoren über die vorgestellten Metriken für die drei Klassen wach, fraglich und müde.

KNN Klassifikator mit k=3

	vorhergesagt wach	vorhergesagt fraglich	vorhergesagt müde
Wahr wach	13	0	0
Wahr fraglich	0	3	6
Wahr müde	0	1	6

Support Vector Machine

	vorhergesagt wach	vorhergesagt fraglich	vorhergesagt müde
Wahr wach	13	0	0
Wahr fraglich	8	1	0
Wahr müde	3	0	4

Logistische Regression

	vorhergesagt wach	vorhergesagt fraglich	vorhergesagt müde
Wahr wach	13	0	0
Wahr fraglich	2	3	4
Wahr müde	1	2	4

Tabelle 4: Confusion Matritzen für die drei Klassifikatoren für die drei Klassen wach, fraglich und müde.

hatten wir 144 Feature-Vektoren, davon jeweils 48 für die Zustände „müde“, „fraglich“ und „wach“, jeweils einmal für jeden Probanden. Um die Feature-Vektoren zu erstellen, haben wir die ersten sechs Minuten jedes Videos für die Kalibrierung genutzt. Die Aufteilung in Test- und Trainingsdatensätze erfolgte zufällig über die 144 Feature-Vektoren, unabhängig von der Person. Durch eine fünf-fache Crossvalidierung haben wir immer 20 Prozent als Test- und 80 Prozent als Trainingsdaten verwendet. Die Werte für „fraglich“ und „müde“ wurden ins Verhältnis zu den Wachwerten gesetzt, und danach wurden die Klassifikatoren trainiert. Dies führte dazu, dass für jedes Feature ein Wach-Wert von 1 vorlag, was die Ergebnisse der Klassifikatoren beeinflusste und bessere Ergebnisse für die verwendeten Metriken lieferte, da alle Testdaten für „wach“ richtig klassifiziert wurden. Trotzdem ergibt die Klassifikation Sinn, da sie eine sinnvolle Trennung zwischen den Klassen ermöglicht und eine signifikante Abweichung von Werten der Klasse „wach“ für die Klassifikation von „fraglich“ und „müde“ aufzeigt.

Die Ergebnisse zeigen im direkten Vergleich der Klassifikatoren für drei Klassen in Tabelle 3:

- Alle Metriken sind bei dem KNN-Klassifikator am besten, wobei die Accuracy und die Precision die höchsten Werte aufweisen.

Klassifikator	F1-Score	Recall	Precision	Accuracy
KNN Klassifikator mit k = 3	1	1	1	0.947368
Logistische Regression	1	1	1	0.904094
Support Vector Machine	0.88797	0.894737	0.908772	0.712865

Tabelle 5: Ergebnisse der drei verglichenen Klassifikatoren über die vorgestellten Metriken für die zwei Klassen wach und müde.

- Bei der Logistic Regression ist die Accuracy höher (0.60) als bei der SVM (0.49), jedoch ist die Precision niedriger (0.67 gegenüber 0.79).
- Der F1 Score und der Recall sind ebenfalls beim KNN-Klassifikator am höchsten.

Ein Blick auf die Confusion Matrizen eines Durchlaufs der Crossvalidierung lohnt sich (siehe Tabelle 4, insbesondere wenn man bedenkt, dass eine Fehlklassifikation von „wach“ oder „fraglich“ als „müde“ in unserem Fall weniger problematisch ist als die Fehlklassifikation von „müde“ als „wach“ oder „fraglich“. Da wir vor Sekundenschlaf warnen möchten, ist eine frühzeitige fehlerhafte Warnung weniger problematisch, natürlich unter der Berücksichtigung, dass zu viele Warnungen als nicht zielgerichtet empfunden werden könnten. Die Confusion Matrix für den KNN-Klassifikator zeigt beispielsweise, dass sechs „fragliche“ Videos als „müde“ klassifiziert wurden, während bei der SVM und der Logistischen Regression eine größere Streuung zu sehen ist: bei der SVM wurden acht Videos als „wach“ statt „fraglich“ klassifiziert, und bei der logistischen Regression zwei als „wach“ statt „fraglich“. Dies ist in unserem Fall ungünstig. Der KNN-Klassifikator hat keine Videos fehlerhaft als „wach“ klassifiziert, sondern lediglich ein Video als „fraglich“ anstatt „müde“. Daher zeigt sich der KNN-Klassifikator als die beste Wahl zur Detektion von Müdigkeit.

Der Vergleich von zwei Klassen führte aufgrund der in Relation gesetzten Werte zu einem noch besseren Ergebnis und dennoch konnte auch hier eine klare Trennung erreicht werden. Wie in Tabelle 5 haben die logistische Regression und der KNN-Klassifikator beide sehr gut abgeschnitten, wobei die Accuracy beim KNN-Klassifikator höher war als bei der logistischen Regression. Dies bedeutet, dass der KNN-Klassifikator in allen Durchläufen eine bessere Klassifikation durchgeführt hat. Ein Beispiel aus einem Kreuzvalidierungsdurchlauf zeigt, dass der KNN-Klassifikator und die logistische Regression in diesem Fall alles richtig klassifiziert haben (siehe Tabelle 6), wie auch die Werte von 1 bei den F1-, Recall- und Precision-Werten in Tabelle 5 zeigen. Die SVM hat in diesem Fall jedoch zwei Werte falsch klassifiziert. Hier ist der Accuracy Score am ausschlaggebendsten.

KNN Klassifikator mit k=3

	vorhergesagt wach	**vorhergesagt müde**
Wahr wach	13	0
Wahr müde	0	6

Support Vector Machine

	vorhergesagt wach	vorhergesagt müde
Wahr wach	13	0
Wahr müde	0	6

Logistische Regression

	vorhergesagt wach	**vorhergesagt müde**
Wahr wach	13	0
Wahr müde	2	4

Tabelle 6: Confusion Matritzen für die drei Klassifikatoren für die zwei Klassen wach und müde.

3.4 Müdigkeitserkennung in der Anwendung

In der Anwendung erfolgt die Kalibrierung des Wachzustands in den ersten 60 Sekunden, in denen die Feature-Vektoren für den Wachzustand berechnet werden. Es sei denn, die Person dreht sich zur Seite oder bewegt sich aus dem Blickfeld, was die Kalibrierung anhält. Wenn die Person für mehr als drei Sekunden nicht nach vorne schaut, ertönt ein Warnton, um den Fahrer zur Konzentration auf die Straße und erhöhten Aufmerksamkeit aufzufordern, da Ablenkung ebenfalls zu Unfällen führen kann.

Nach der Kalibrierung wird für jedes Einzelbild der aktuelle Feature-Vektor neu berechnet und ins Verhältnis zu den Wachwerten gesetzt. Wenn der KNN-Klassifikator eine signifikante Abweichung von den Wachwerten feststellt, ändert sich die Farbe um das Video herum, zuerst zu Orange („fraglich“) und anschließend zu Rot („müde“). Um abrupte Änderungen, wie sie in unseren Tests auftraten, auszugleichen, wird der Medianwert (wach/fraglich/müde) der letzten 60 Sekunden als maßgebend betrachtet.

Wenn eine Person über einen längeren Zeitraum im müden Zustand verbleibt, erfolgt eine wiederholte Warnung. Dies geschieht alle zehn Sekunden, um eine angemessene Häufigkeit sicherzustellen. Zusätzlich wird eine Warnung ausgegeben, wenn eine Person unabhängig von ihrem Status die Augen zu lange geschlossen hält, in unserem Fall beträgt dies genau zwei Sekunden.

Unsere Tests haben gezeigt, dass unser Müdigkeitsdetektionskonzept eine solide Grundlage bietet, jedoch die verschiedenen Parameter nicht für jede Person optimal sind. Es

gibt auch feststellbare Abweichungen in Bezug auf den Zeitpunkt, zu dem Warnungen ausgegeben werden, was vermutlich auf gelegentliche Berechnungsfehler zurückzuführen ist. Dies sind normale Schwankungen, die in Zukunft genauer betrachtet werden sollten. Trotzdem funktioniert der Übergang zwischen den verschiedenen Zuständen, auch wenn man sich in manchen Situationen nur sehr kurz im Zustand „fraglich“ befindet.

4 Ausblick

Während des gesamten Erstellungsprozesses sind wir als Gruppe vor einige Herausforderungen geraten, an welchen wir stets wachsen und lernen konnten. Dieses Kapitel dient als eine Reflektion unserer Arbeitsweise, in dem wir aufzeigen, was für zukünftige Projekte erlernt wurde, um das eigene Vorgehen stets verbessern zu können. Weiterhin soll eine Darstellung möglicher Erweiterungen die Vielfältigkeit des gewählten Themengebietes aufzeigen und Inspiration für weitere Arbeiten geben.

4.1 Lessons Learned

Während des gesamten Erstellungsprozesses sind uns als Gruppe einige Punkte bewusst geworden, welche wir bei einer nächsten Zusammenarbeit gesondert beachten würden. Der erste Punkt ist in der Planungsphase anzusiedeln, genauer in der zeitlichen Einteilung. Uns wurde im Laufe der Entwicklungen bewusst, dass mehr Zeit für Fehler und zu behebende Probleme eingeplant werden muss. Wir planten zu optimistisch und rechneten mit deutlichen weniger Schwierigkeiten, was uns bedingt in Stress verfallen lassen hat. Bei der nächsten Planung ist demzufolge nicht zu optimistisch heranzugehen, es muss mehr Kapazität für unerwartete Fehler eingeplant werden, welche nicht in kürzester Zeit behoben werden können, um trotzdem noch den zeitlichen Rahmen einhalten zu können.

Wird sich tiefer in die Müdigkeitserkennung eingearbeitet, so fällt schnell die Vielfältigkeit des Themas auf. Ideen, die anfangs gesammelt wurden und umgesetzt werden sollten, sind nach ausführlicher Recherche nur noch ein kleiner Teil eines großen Kontingents an Möglichkeiten. Davon haben wir uns anfangs stark beeinflussen lassen und wir wollten jede Möglichkeit austesten, um teilweise auch nur marginale Verbesserungen zu erreichen. Später ist uns die Relevanz eines klar definierten Fokus bewusst geworden. Hierbei sollten bei der Recherche und dem Prüfen verschiedener Erweiterungen stets Aufwand und Nutzen gegenübergestellt werden. Da wir dies zuerst missachteten, ver-

brachten wir viel Zeit damit, verschiedene Methoden zu prüfen anstatt sich gemeinsam festzulegen und dies weiter zu vertiefen. Um den Zeitplan und Umfang des Projektes einzuhalten, ist demzufolge stets ein klarer Fokus und eine Betrachtung des Aufwands in Relation zum Nutzen nötig.

Eine weitere wichtige Erkenntnis, die wir gewonnen haben, war die Bedeutung einer umfassenden Betrachtung aller Faktoren, bevor wir uns auf eine bestimmte Technologie festlegen. In unserem Fall war es ein Fehler, sich frühzeitig auf Python als Programmiersprache festzulegen, ohne die Alternativen zu prüfen. Eine gründliche Analyse aller relevanten Faktoren, einschließlich der Eignung der Programmiersprache, hätte uns möglicherweise zu besseren Entscheidungen geführt. Wir haben nach Abschluss des Projekts erkannt, dass Java und Kotlin als Programmiersprachen besser zu unseren Anforderungen gepasst hätten. Beide Sprachen bieten eine reibungslose Integration der benötigten Bibliotheken wie OpenCV und MediaPipe. Diese Erkenntnis hat uns gezeigt, wie wichtig es ist, die Flexibilität bei der Wahl der Technologie zu wahren und solche Optionen offen zu halten.

Der nächste wesentliche Lernpunkt war die Priorisierung von Design und Usability. In unserem Fall lag die Hauptaufmerksamkeit zu stark auf der reinen Integration der Müdigkeitserkennungslogik und dem Versuch des Deployments der App. Dabei hätten wir mehr Ressourcen und Aufmerksamkeit auf das Design und die Benutzerfreundlichkeit der App und des Interfaces lenken können.

Darüber hinaus haben wir erkannt, dass die App-Entwicklung noch tiefer gehen kann, insbesondere im Hinblick auf die Nutzung der Hardware des Endgeräts. Wir könnten spezielle Techniken zur Verbesserung der Stabilität und Geschwindigkeit der App erforschen. Dies umfasst die effiziente Nutzung von Ressourcen wie Kamera und Prozessor, um eine reibungslose Erfahrung für die Benutzer sicherzustellen.

Insgesamt haben diese Herausforderungen und Lernprozesse unser Verständnis für die Entwicklung von Apps erweitert und uns dabei geholfen, fundierte Entscheidungen zu treffen. Diese Erkenntnisse werden uns in zukünftigen Projekten dabei helfen, die Qualität unserer Apps weiter zu verbessern und die Anforderungen der Benutzer optimal zu erfüllen.

Eine der herausforderndsten Phasen unseres Projekts war zweifellos der Versuch, die App als APK-Datei bereitzustellen. Diese Herausforderungen ergaben sich aus unserer anfänglichen Entscheidung, Python und das Kivy-Framework zu verwenden, was sich

als ein limitierender Faktor für das Deployment herausstellte. Ein bedeutendes Problem war die Unausgereiftheit des Build-Prozesses mit Kivy im Vergleich zu etablierten Alternativen wie Android Studio in Verbindung mit Java oder Kotlin. Der Build-Prozess mit Kivy war anfälliger für Fehler und Schwierigkeiten. Je mehr Abhängigkeiten und Bibliotheken in unserer Logik verwendet wurden, desto geringer war die Wahrscheinlichkeit, dass der Build-Prozess ohne Fehler durchlief.

Wir haben schmerzlich erfahren, dass die Wahl von Kivy für unser Projekt die Möglichkeit, die App als APK-Datei bereitzustellen, stark eingeschränkt hat. Dies führte zu erheblichem Zeitaufwand und frustrierenden Versuchen, Lösungen zu finden. Wir unternahmen eine Vielzahl von Rettungsversuchen, darunter das Deployment über Docker und die Implementierung in einem Container. Wir gingen sogar so weit, die grundlegende Funktionsweise von Kivy beim Kompilieren und Builden tiefgehend zu analysieren. Die Zeit, welche in diese Bemühungen investiert wurde, hätte stattdessen genutzt werden können, um die Potenziale zu erforschen, die bereits im Kapitel 2.8.3 erwähnt wurden. Eine verstärkte Fokussierung auf die Verbesserung des Designs, der Usability und der Effizienz der App sowie die tiefere Integration der Hardware des Endgeräts hätte unsere App insgesamt verbessern können.
Diese Deployment-Herausforderungen haben uns wertvolle Einsichten vermittelt, darunter die Notwendigkeit, den Build-Prozess frühzeitig zu evaluieren und eine gut durchdachte Wahl der Entwicklungsumgebung zu treffen. Sie haben uns auch gezeigt, wie wichtig es ist, die Abhängigkeiten und Anforderungen der ausgewählten Technologie sorgfältig zu berücksichtigen, um unerwartete Schwierigkeiten im späteren Entwicklungsprozess zu vermeiden.

4.2 Future Work

Um die Vielfältigkeit dieses Projektes aufzuzeigen und für weitere Arbeiten zu inspirieren, ist es sinnvoll, noch offene Themen und Erweiterungen zu beleuchten. Der erste nennenswerte Punkte ist ein gezieltes Erhöhen der Robustheit des Programms. Der für uns optimale Schwellwert zur Blinzeldetektion wurde empirisch ermittelt und war für unser Anwendungsgebiet passend. Sollte man diese Voreinstellungen auf andere Ethnien testen, könnte es jedoch gegebenenfalls zu Komplikationen kommen. Um dem entgegenzuwirken, sollte der Schwellwert variabel gesetzt werden, indem ein optimaler relativer Threshold zum durchschnittlichen EAR-Wert der Person ermittelt wird. Um diesen zu ermitteln sind jedoch weitere Test und Forschung notwendig, jedoch ist dies eine Verbesserung der Anwendbarkeit des Programms.

Weiterhin kann die Wahl anderer Klassifikationsverfahren eine zusätzliche Genauigkeitsverbesserung mit sich bringen. Hierfür sind verschiedenste Versuche notwendig, um aus vielen Klassifikatoren den bestgeeigneten ermitteln zu können, wir haben uns dabei aus aufwandsmindernden Gründen auf drei Möglichkeiten beschränkt. Die Wahl einer anderen Methode könnte weiterhin den Vorteil mit sich bringen, dass man unter zusätzlicher Vergrößerung des Datensatzes einen Score zur Müdigkeit entwickeln kann. Somit verfällt eine strikte Einteilung in vordefinierte Klassen und Veränderungen der Werte über bestimmte Zeitreihen sind besser nachzuvollziehen und zu analysieren.

Um unser Anwendungsgebiet des Autofahrens weiter zu vertiefen und die Sicherheit zusätzlich zu erhöhen, ist es möglich, nicht nur die Müdigkeit zu erkennen, sondern auch Unaufmerksamkeiten zu detektieren. Von hoher Relevanz hierbei kann ein Erkennen der Zeit sein, in welcher der Fokus nicht auf den Verkehr gelegt wird. Somit erscheint eine Warnung, wenn sich der Blick zu lange dem Handy, Infotainmentsystem des Fahrzeugs oder in der weniger relevanten Fahrumgebung richtet.

Sollte eine solche Software professionelle Anwendung in einem Auto finden, so ist ein Einsatz auf sich im Auto befindender Hardware unausweichlich. Somit kann beispielsweise im Armaturenbrett eine kleine Webcam installiert sein, welche über das Boardsystem angeschlossen ist und die Warnungen können direkt über ein Cockpit, Head-Up Display oder die interne Soundanlage wiedergegeben werden. Sollten diese Schritte zukünftig genauer bedacht werden, so würde dem Weg dieser Technologie in Serienfahrzeuge nichts entgegenstehen und die Sicherheit aller Verkehrsteilnehmenden kann somit maßgeblich verbessert werden.

4.3 Zusammenfassung

Zusammenfassend lässt sich für die Projektarbeit sagen, dass im Rahmen des gesamten Erarbeitungsprozesses viel gelernt wurde. Einerseits konnte theoretisches Fachwissen in die Praxis umgesetzt werden, andererseits konnte man erste Erfahrungen mit der Planung und Organisation einer solchen Entwicklung sammeln. Unsere Anfangsvorstellung einer Anwendung zur Müdigkeitsdetektion über eine interne Kamera konnten wir in die Realität umsetzen und, weiter verbessert und optimiert, kann sie einen maßgeblichen Einfluss auf die Verkehrssicherheit nehmen. Wir betiteln sowohl dieses Modul, als auch unsere Gruppenarbeit als erfolgreich und sind mit unserem Ergebnis zufrieden, unter der Prämisse, dass dies ein nie endendes Thema mit stetiger Weiterentwicklung ist. Mit unserer Arbeit ist der Grundstein für mögliche weitere Arbeiten gegeben.

Literatur

[AG90] Akerstedt, T. ; Gillberg, M.: Subjective and objective sleepiness in the active individual. In: *The International journal of neuroscience* 52 (1990), Nr. 1-2, S. 29–37. http://dx.doi.org/10.3109/00207459008994241. – DOI 10.3109/00207459008994241. – ISSN 0020–7454

[AYA+20] Arunasalam, M. ; Yaakob, N. ; Amir, A. ; Elshaikh, M. ; Azahar, N. F.: Real-Time Drowsiness Detection System for Driver Monitoring. In: *IOP Conference Series: Materials Science and Engineering* 767 (2020), Nr. 1. http://dx.doi.org/10.1088/1757-899X/767/1/012066. – DOI 10.1088/1757–899X/767/1/012066. – ISSN 1757–8981

[BBC+97] Bentivoglio, A. R. ; Bressman, S. B. ; Cassetta, E. ; Carretta, D. ; Tonali, P. ; Albanese, A.: Analysis of blink rate patterns in normal subjects. In: *Movement disorders : official journal of the Movement Disorder Society* 12 (1997), Nr. 6, S. 1028–1034. http://dx.doi.org/10.1002/mds.870120629. – DOI 10.1002/mds.870120629. – ISSN 0885–3185

[DCC+22] Dewi, Christine ; Chen, Rung-Ching ; Chang, Chun-Wei ; Wu, Shih-Hung ; Jiang, Xiaoyi ; Yu, Hui: Eye Aspect Ratio for Real-Time Drowsiness Detection to Improve Driver Safety. In: *Electronics* 11 (2022), Nr. 19, S. 3183. http://dx.doi.org/10.3390/electronics11193183. – DOI 10.3390/electronics11193183

[DT05] Dalal, N. ; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Schmid, Cordelia (Hrsg.): *Proceedings*, IEEE Computer Society, 2005. – ISBN 0–7695–2372–2, S. 886–893

[GGA] Ghoddoosian, Reza ; Galib, Marnim ; Athitsos, Vassilis: *A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection.* http://arxiv.org/pdf/1904.07312v1

[Kin09] King, Davis E.: Dlib-ml: A Machine Learning Toolkit. In: *Journal of Machine Learning Research* 10 (2009), 1755-1758. http://dlib.net/face_landmark_detection.py.html

[LTN+] Lugaresi, Camillo ; Tang, Jiuqiang ; Nash, Hadon ; McClanahan, Chris ; Uboweja, Esha ; Hays, Michael ; Zhang, Fan ; Chang, Chuo-Ling ; Yong, Ming G. ; Lee, Juhyun ; Chang, Wan-Teh ; Hua, Wei ; Georg, Manfred ; Grundmann, Matthias: *MediaPipe: A Framework for Building Perception Pipelines.* https://arxiv.org/pdf/1906.08172

[Mar] Mariella Dreissig, Mohamed Hedi Baccour, Tim Schäck, Enkelejda Kasneci: Driver Drowsiness Classification Based on Eye Blink and Head Movement Features Using the k-NN Algorithm.

[Par16] Parisa Ebrahim: Driver drowsiness monitoring using eye movement features derived from electrooculography: Doktorarbeit. (2016)

[PVG+11] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[RKA+19] Ramzan, Muhammad ; Khan, Hikmat U. ; Awan, Shahid M. ; Ismail, Amina ; Ilyas, Mahwish ; Mahmood, Ahsan: A Survey on State-of-the-Art Drowsiness Detection Techniques. In: *IEEE Access* 7 (2019), S. 61904–61919. http://dx.doi.org/10.1109/ACCESS.2019.2914373. – DOI 10.1109/ACCESS.2019.2914373

[Sou16] Soukupova, Tereza: Eye-Blink Detection Using Facial Landmarks. In: *Research Reports of CMP* 5 (2016)

[WWK+94] Wierwille, Walter W. ; Wreggit, Steven S. ; Kirn, Chris ; Ellsworth, Lynne A. ; Fairbanks, Rollin J.: Research on vehicle-based driver status/performance monitoring; Development, validation, and refinement of algorithms for detection of driver drowsiness. Final report, 1994

Abbildungsverzeichnis