

Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik

IoT - Smart School v2

Eingereicht von: **Simon Danninger, 5AHIF**

Christoph Pfleger, 5AHIF

Datum: **April 5, 2020**

Betreuer: **Dipl. Ing. Herbert Lackinger**

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 5. April 2020

Simon Danner, Christoph Pfleger

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 5, 2020

Simon Danner, Christoph Pfleger

Gender Klausel

Zur besseren Lesbarkeit wird in der vorliegenden Arbeit auf geschlechterspezifische Formulierung verzichtet. Es wird ausdrücklich darauf hingewiesen, dass im Text personenbezogenen Formulierungen immer gleichermaßen auf alle Geschlechter bezogen sind.

Gender Clause

For better readability, gender-specific wording is not included in this paper. It is expressly pointed out that the wording in the text refers to all genders in the same way

Zusammenfassung

IoT - Smart School ist ein Projekt, welches das Ziel verfolgt, die Lebensqualität in Klassenräumen, bei gleichzeitiger Kostenreduktion, zu verbessern. Das Projekt liefert Information zur Luftqualität in Klassenräumen und macht Vorschläge um die Luft zu verbessern. Dies bietet einen Vorteil für die Schüler. Durch unterschiedliche Optimierungen, wie Beamer ausschalten oder richtiges Lüften, können Heiz- und Stromkosten gesenkt werden. Eine künstliche Intelligenz kann aus vorhandenen Daten den Status der Fenster in einem Raum, offen oder geschlossen, vorhersagen. Mit einem live Video Feed können Veranstaltungen in allen Klassen angeschaut werden. Messwerte aus der Vergangenheit können aufbereitet und in Grafiken veranschaulicht werden. Dieses Projekt bildet den Grundstein eines intelligenten Smart Home - beziehungsweise Smart School - System und ermöglicht in Zukunft die weitere Implementierung neuer Features.

Abstract

IoT - Smart School is a project that aims to improve the quality of life in classrooms while reducing costs. The project provides information on air quality in classrooms and makes suggestions to improve air quality. This offers an advantage for the students. By various optimizations, such as switching off the beamer when nobody is in the room or proper ventilation, heating and electricity costs can be reduced. Artificial intelligence can predict the status of the windows in a room, open or closed, from existing data. With a video stream every student can watch events in every class. Measured values from the past can be prepared and illustrated in graphics. This project forms the cornerstone of an intelligent Smart Home or School system and will enable the further implementation of new features in the future.

Danksagung

An dieser Stelle möchten wir uns bei all jenen bedanken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.

Besonders möchten wir uns auch bei Herrn Prof. Lackinger bedanken, welcher uns durch seine freundliche Hilfsbereitschaft bestens unterstützt hat.

Inhaltsverzeichnis

1 Einleitung	10
1.1 Anfangssituation	10
1.2 Zielsetzung	11
2 Theorie	12
2.1 Streaming [SD]	12
2.1.1 mjpeg	12
2.1.1.1 Vorteile	12
2.1.1.2 Nachteile	13
2.1.2 ffmpeg	13
2.1.2.1 Vorteile	13
2.1.2.2 Nachteile	14
2.1.3 Adobe RTMP	14
2.1.3.1 Vorteile	14
2.1.3.2 Nachteile	15
2.1.4 WebRTC	15
2.1.4.1 Vorteile	15

2.1.4.2	Nachteile	16
2.1.4.3	TURN	16
2.1.4.4	STUN	16
2.2	Raspberry [SD]	17
2.2.1	Funktion	17
2.2.2	Vorteile	18
2.2.3	Nachteile	18
2.2.4	Alternativen	19
2.2.4.1	Banana Pi	19
2.2.4.2	Arduino	19
2.2.5	Raspberry Zero	19
2.2.6	Motion	20
2.3	Node JS [SD]	21
2.3.1	Vorteile	21
2.3.2	Nachteile	22
2.3.3	Alternative: Go	23
2.3.3.1	Vorteile von Go	23
2.3.3.2	Nachteile von Go	23
2.3.4	Alternative: Asp.Net	24
2.3.4.1	Vorteile von Asp.Net	24
2.3.4.2	Nachteile von Asp.Net	24
2.4	npm [SD]	25
2.4.1	Alternativen	25

2.5	Angular [SD]	26
2.5.1	Funktion	26
2.5.2	Vorteile	26
2.5.3	Nachteile	27
2.5.4	Alternative React	27
2.5.5	Angular Material	27
2.6	Three-js [SD]	28
2.6.1	Vorteile	28
2.6.2	Nachteile	28
2.7	Postgres [SD]	29
2.8	MQTT [SD]	29
2.9	Http [SD]	30
2.9.1	Https	30
2.9.2	Tls Ssl	30
2.10	Rest [SD]	32
2.11	Git [SD]	33
2.12	Docker [SD]	34
2.12.1	Grundbegriffe	34
2.12.2	Docker-compose	35
2.12.3	Traefik	35
2.13	Künstliche Intelligenz - Machine Learning [CP]	36
2.13.1	Mathematische Notation	37
2.13.1.1	Skalar	37

2.13.1.2	Vektor	37
2.13.1.3	Matrix	38
2.13.1.4	Funktionen	38
2.13.2	Arten	38
2.13.2.1	Supervised Learning	39
2.13.2.2	Unsupervised Learning	39
2.13.2.2.1	Clustering	40
2.13.2.2.2	Dimensionsreduktion	40
2.13.2.2.3	Außenseitererkennung	40
2.13.2.3	Semi-Supervised Learning	40
2.13.2.4	Reinforcement Learning	41
2.13.3	Lineare Regression	41
2.13.3.1	Gradient Descent	43
2.13.4	Logistische Regression	45
2.13.5	Neuronale Netzwerke	47
2.13.5.1	Forward Feeding	49
2.13.5.2	Back Propagation	50
2.13.5.3	Aktivierungsfunktionen	50
2.13.5.4	Recurrent Neural Networks	50
2.14	Python [CP]	52
2.14.1	Vorteile	52
2.14.2	Nachteile	52
2.14.3	Alternative	53

2.14.3.1	C++	53
2.15	Numpy [CP]	53
2.15.1	Vorteile	53
2.16	Jupyter Lab [CP]	53
2.17	TensorFlow [CP]	54
2.17.1	Graphen	54
2.17.2	Keras	55
2.17.2.1	Layers	55
2.17.2.2	Vorteile	56
2.17.2.3	Nachteile	56
2.17.2.4	Alternative	56
2.17.3	Vorteile	56
2.17.4	Nachteile	57
2.17.5	Alternative	57
2.17.5.1	PyTorch	57
2.18	JSON [CP]	57
3	Praktischer Teil	59
3.0.1	Aufbau [SD]	59
3.0.2	Bestandteile[SD]	60
3.1	Raspberry Camera [SD]	61
3.1.1	Aufgabenstellungen	61
3.1.2	Aufgabenstellung: Webcam - Stream	61

3.1.2.1	WebRTC	61
3.1.3	Motion	62
3.1.4	Aufgabenstellung: Webcam ein - und ausschalten	63
3.1.5	Sicherheit	63
3.1.6	Konfiguration	64
3.1.6.1	Python-Controller	64
3.1.6.1.1	Raum	64
3.1.6.1.2	Mqtt Broker	64
3.1.6.1.3	SSL Verschlüsselung	64
3.1.6.1.4	Motion	65
3.1.6.2	NGINX	66
3.1.7	Bedienung	67
3.1.7.1	Python Version	67
3.1.7.2	Automatischer Start beim Hochfahren	68
3.1.7.2.1	Systemd	68
3.2	IoT - Web [SD]	69
3.2.1	Aufgaben	69
3.2.2	3d Model	69
3.2.3	IoT - Web Core	73
3.2.4	Dashboard	73
3.3	Installation [SD]	74
3.3.1	Raspberry Pi Video Stream	74
3.3.1.1	Raspberry Installation	74

3.3.1.2	Installation vom Betriebssystem	74
3.3.1.3	Raspberry Pi Aufsetzen	75
3.3.1.4	Python	75
3.3.1.5	Venv	75
3.3.1.6	SSL Zertifikat	75
3.3.1.7	Motion	76
3.3.1.8	Nginx	76
3.3.1.9	NGINX Sicherheit:	77
3.3.2	Installation auf der Virtuellen Maschine	77
3.3.2.1	Docker - Compose	77
3.4	System Architektur - Künstliche Intelligenz [CP]	78
3.4.1	Datensammlung	80
3.4.2	Installieren und Starten	82
3.5	Künstliche Intelligenz [CP]	83
3.5.1	Vorgehensweise	83
3.5.1.0.1	Modell definieren	83
3.5.1.0.2	Daten einlesen	83
3.5.1.0.3	Daten formatieren	83
3.5.1.0.4	Daten aufteilen	83
3.5.1.0.5	Modell trainieren	84
3.5.1.0.6	Bewerten und Verändern	84
3.5.1.0.7	Evaluieren	84
3.5.1.0.8	Modell speichern	84

3.5.2	Modell	84
3.5.2.1	Metriken	86
3.5.2.1.1	Confusion Matrix	86
3.5.2.1.2	Accuracy	86
3.5.2.1.3	Precision	87
3.5.2.1.4	Recall	87
3.5.3	Datenverwaltung	88
3.5.3.1	Einlesen	88
3.5.3.2	Normalisieren	89
3.5.3.3	Aufteilen	91
3.5.3.4	Umsetzung	93
3.5.4	Training	93
3.5.5	Bewertung	98
3.5.5.1	Prototyping Phase	98
3.5.5.2	Evaluation	99
3.5.6	Abspeichern	101
3.6	MQTT Clients [CP]	101
3.6.1	Automatischer WakeUp	101
3.6.2	Künstliche Intelligenz	102
4 Fazit und Ausblick		103

Kapitel 1

Einleitung

1.1 Anfangssituation

Aufgrund der tendenziell fallenden Luftqualität in den Klassenräumen, welche zu schlechterer Lernqualität der Schüler führt, wurde im Jahr 2019 das IoT - Smart School Projekt gestartet. Dieses sollte die Luftqualität der Klassen messen und Schüler auf schlechte Luftqualität aufmerksam machen. Falsches oder dauerhaftes Lüften kann die Heizkosten unnötig erhöhen. Es kommt auch häufiger vor, dass in Klassenräumen vergessen wurde die Fenster zu schließen und diese längere Zeit offen stehen, sofern der Schulwart oder etwaige Reinigungskräfte diese nicht schließen. Aufgrund des überfüllten Buffet kann es dazu kommen das die kleine Pause der Schüler nicht ausreicht um sich Essen oder Trinken zu kaufen. Bei größeren Veranstaltungen kann der Festsaal schnell zu klein werden, und somit müssen diese Veranstaltungen entweder öfter abgehalten werden oder Schüler haben keine Möglichkeit sich diese anzusehen.

1.2 Zielsetzung

Da das Ausstatten der über 500 Fenster der Schule mit Öffnungskontakten aus Kostengründen nicht möglich ist, soll eine Künstliche Intelligenz anhand bereits vorhandener Sensorwerten den Status der Fenster in einer Klasse, offen oder geschlossen, mit einer Genauigkeit von über 80% bestimmen. Für die Zeitersparnis am Buffet soll die Möglichkeit bestehen über einen Live - Videostream die aktuelle Warteschlange des Buffets beobachten zu können. Auch für die “Vergrößerung” des Festsaales in den Online Bereich kann ein Videostream verwendet werden.

Das Gesamtsystem soll weiterhin in der Praxis eingesetzt und gewartet werden. Wie auch in diesem Jahr soll dieses Projekt in den Folgejahren übernommen und weiterentwickelt werden.

Kapitel 2

Theorie

2.1 Streaming [SD]

2.1.1 mjpeg

Motion JPEG setzt für einen Videostream auf einzeln kodierte und dekodierte JPEG. Diese Kodierung und Dekodierung erfolgt in Echtzeit. Dabei kann Motion JPEG auch Audio Video Interleave übertragen. Man spricht von Audio Video Interleave, wenn die Ton- und Audiospur ineinander verkettet sind. Gängige RGB und YUV-Formate werden unterstützt. YUV ist ein älteres Format für die Darstellung eines Bildes, anstatt wie RGB auf die einzelnen Farbkanäle zu setzen, verwendet es die Lichtstärke (Luminanz) und die Farbanteile (Chrominanz). Durch MIME multipart/x-mixed-replace;boundary, was soviel heißt wie, dass mehrere Teile getrennt durch ein bestimmtes Zeichen gesendet werden, kann der Stream über eine HTTP-Verbindung übertragen werden.

2.1.1.1 Vorteile

- Keine Kompression bei der Übertragung.
Da die Daten nicht sonderlich weiter als die übliche JPEG-Komprimierung verkleinert werden, verliert man nicht an Bildqualität.

- Verfügbarkeit auf allen Webbrowsern.

Durch die HTTP - Übertragung ist Motion JPEG in allen Browsern verfügbar.

2.1.1.2 Nachteile

- Verbraucht größere Bandbreite als Alternativen.
- Hohe CPU/GPU Auslastung durch Ständiges dekodieren oder encodieren.
- Kein Feedback seitens des Clients möglich.

2.1.2 ffmpeg

Ffmpeg ist eine Cross-Platform Lösung, welche Videos und Audio aufnehmen, konvertieren und auch streamen kann. Sie ist gratis und Open-Source.

Entwickelt wurde ffmpeg ursprünglich für die Verwendung in einer Command Line. Einsatzbereiche sind sowohl einfache Videobearbeitungen sowie verkürzen oder zusammenhängen von Videos. Außerdem ist es möglich das Video zu skalieren, also die Größe beziehungsweise die Auflösung zu verändern. Weitere Optionen sind die Vertonung von Videos oder das Einfügen von Untertiteln.

Selbst professionelle Videobearbeitungsmöglichkeiten wie Korrektur von Helligkeit, Kontrast, Gamma, Stabilisierung von verwackelten Videos und weiterem ist möglich.

2.1.2.1 Vorteile

- Großer Umfang an Möglichkeiten.
- Konvertierung von unterschiedlichsten Formaten.
- Die Auflösung spielt keine Rolle.
- Aktiver Mail-Support.

2.1.2.2 Nachteile

- Langer Zeitraum um mit dem Tool vertraut zu werden.
- Dokumentation ist nicht aktuell.
- Es gibt in der Dokumentation fast keine Beispiele.

2.1.3 Adobe RTMP

Adobe Real Time Messaging Protocol ist ein proprietäres Protokoll welches Adobe gehört. Ursprünglich wurde es von Macromedia entwickelt und später an Adobe verkauft. Es verwendet für die Übertragung TCP.

Transmission Control Protocol ist ein Standard, welcher beschreibt wie Daten über ein Netzwerk ausgetauscht werden. Es deckt sowohl fehlerhafte Übertragungen ab, als auch die Aufrechterhaltung der Verbindung durch regelmäßige Übertragung von Kontrolldaten.

Entwickelt wurde RTMP für die Kommunikation zwischen Adobe Flash Technologien, wie Adobe Flash Player oder Adobe AIR. Es bietet eine Verbindung zur Übertragung von Video, Audio und Text. Beide Geräte sind dabei Sender als auch Empfänger von Daten. Für die Übertragung können viele beliebten Übertragungsformate wie AMF (Additive manufacturing file format), SWF Shockwave, FLV Flash Video verwendet werden.

2.1.3.1 Vorteile

- Jedes Format kann übertragen werden.
RTMP beschreibt nur wie die Daten übertragen werden. Dabei ist es Format unabhängig.
- Verschlüsselung möglich.
- Geringe Verzögerung bei der Übertragung.
- Support für ältere Geräte die kein HTTP verstehen.
Ältere Geräte verstehen teilweise noch kein HTTP aber TCP und dadurch kann auf diesen älteren Geräten RTMP eingesetzt werden.

2.1.3.2 Nachteile

- Im Web nur mit Adobe Flash verwendbar.
- Firewallprobleme.
Adobe RTMP verwendet nicht den Standardport 80 welcher von HTTP verwendet wird, sondern die PORT 443 und 1935. Der Port 1935 kann von manchen Firewalls blockiert werden.
- Neue Content Delivery Networks unterstützen diese Technologie nicht mehr. Content Delivery Networks oder auch Content Distribution Networks sind Netzwerke, welche regional verteilt sind aber als gemeinsamer Server agieren. Insbesondere für große Dateien sind diese Netzwerke ideal.

2.1.4 WebRTC

WebRTC steht für Web Real Time Communication also Echtzeit-Kommunikation im Web, es ist ein offener Standard, der aus Kommunikationsprotokollen besteht und durch Programmierschnittstellen oder kurz API definiert ist. Es kann Video & Audio - Stream und auch generische Daten senden und empfangen. Ein weiteres Feature ist das sogenannte Broadcasting, dabei gibt es einen Sender, welcher Daten versendet und beliebig viele Empfänger, welche diese Daten empfangen können. Durch seine JavaScript API kann es auf, beziehungsweise in, jedem modernen Browser laufen. WebRTC stammt von Google und wird auch von Google betreut. Das Projekt ist aber Open-Source.

2.1.4.1 Vorteile

- Platformunabhängig
- Verschlüsselt
- Open-Source

2.1.4.2 Nachteile

- Kann von Firewalls blockiert werden.
- STUN/TURN - Server können notwendig sein.

2.1.4.3 TURN

STUN: Protokoll um die öffentliche IP-Adresse zu ermitteln. Diese wird für einen Verbindungsaufbau oft benötigt.

2.1.4.4 STUN

TURN: ein Server im Internet der Daten zwischen 2 Clients austauschen kann, wenn diese keine direkte Verbindung aufbauen können.

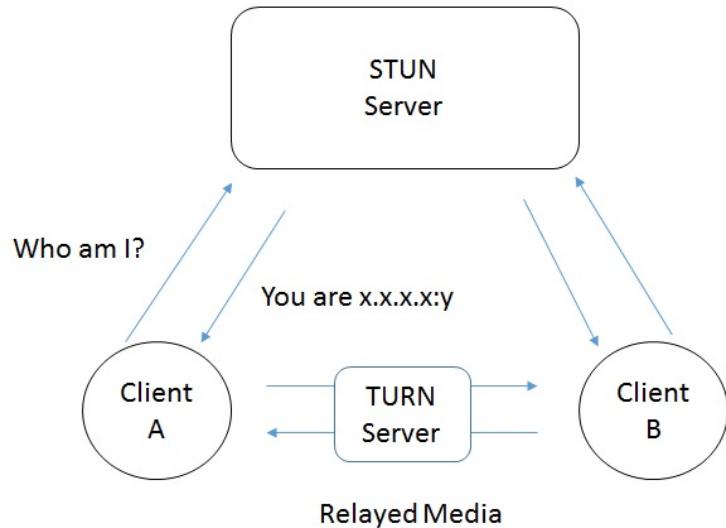


Abbildung 2.1: Erklärung von Stun & Turn [14]

2.2 Raspberry [SD]

Raspberry Pi ist ein Einplatinencomputer, welcher von der britischen Firma Raspberry entwickelt und verkauft wird und nur etwas Größer ist als eine Scheckkarte. Um diese Kompaktheit zu erreichen, setzt Raspberry auf eine Advanced RISC Machines ARM - Architektur. Welche auch in Smartphones oder anderen mobilen Geräten gebräuchlich ist. Entwickelt wurde dieser Minicomputer, um jungen Menschen Programmier- und Hardwarekenntnisse näherzubringen. Der erste Raspberry kam 2012 auf den Markt. 2019 kam die vierte Generation des Raspberry, der Raspberry Pi 4, auf dem Markt. Aufgrund diverser Erweiterungsmöglichkeiten und Softwarepakete ist der Raspberry Pi gerade im Do - It - Yourself Bereich sehr beliebt.



Abbildung 2.2: Raspberry Pi 4 [7]

2.2.1 Funktion

In dieser Diplomarbeit wurde der Raspberry als Streaming Quelle verwenden. Erweitert wurde der Raspberry 3 dafür mit einer USB-Webcam. Mittels Motion, einem Softwarepaket für Livestreaming, und einem Python Script wurde der Raspberry zu einem Livestream Server. Dieser kann ohne weitere Konfiguration in jedes Netzwerk integriert werden. Alles was der Raspberry braucht, ist ein Stromanschluss und eine Internetverbindung. Der Raspberry kann auf mehreren Empfängern gleichzeitig streamen. Hat ein Raspberry gerade keinen Zuseher schaltet er auch die Aufnahme ab.

2.2.2 Vorteile

- handlich

Aufgrund seiner geringen Größe und Gewicht kann der Raspberry an so gut wie jeder Stelle verwendet oder installiert werden.

- effizient

Durch die ARM-CPU ist der Raspberry Pi sehr energieeffizient und verbraucht gerade einmal 4 Watt.

- billig

Für 40 € erhält man das Einstiegsmodell der neuesten Generation.

- erweiterbar

Es gibt unzählige physische Erweiterungen für den Raspberry, wie das Raspberry Kameramodul oder Steckplatinen auf der man dann eigene Schaltungen bauen kann.

- vielseitig

Ein Raspberry kann von einfachen Projekten wie einem Netzwerkspeicher bis hin zu komplexen Bereichen wie Heiamtomaticierung (Smart Home) oder als ein Smart TV eingesetzt werden.

2.2.3 Nachteile

- Leistung

In punkto Leistung kann ein Raspberry mit einem modernen Smartphone nicht mithalten, geschweige denn mit einem handelsüblichen Computer oder Laptop.

- Grundaustattung

Für einen umfangreichen Einsatz müssen diverse Erweiterungen erworben werden, da der Raspberry im Lieferumfang eher nur als Server eingesetzt werden kann. USB Erweiterungen sind zwar verfügbar aber funktionieren unter Umständen nicht so gut wie direkt erworbene proprietäre Module.

- Kein Windows

Ältere Windowsversionen werden aufgrund der ARM - Architektur nicht unterstützt und für neue Windowsversionen welche mit ARM kompatibel sind, reicht die Power des kleinen Rechners nicht aus.

- kein Plug and Play

Der Raspberry kommt ohne Speichermedium oder vorinstallierter Software, diese muss vom Benutzer installiert werden.

2.2.4 Alternativen

2.2.4.1 Banana Pi

Der Name Banana Pi lässt fast schon vermuten, dass es sich hier um einen Konkurrenten im gleichen Segment handelt. Dieser bietet sehr ähnliche Möglichkeiten wie ein Raspberry Pi. Er war bis zum Erscheinen des Raspberry Pi 4 auch leistungsmäßig stärker und hatte mehr RAM. Die Entscheidung fiel, aufgrund der Vertrautheit mit dem Raspberry und das bereits ein Raspberry vorhanden war, auf diesen.

2.2.4.2 Arduino

Ein Arduino ist kein Mini Computer wie der Raspberry, sondern eine Mikroplatine. Eine Mikroplatine kann zwar Code ausführen doch da es kein direktes Betriebssystem enthält, passiert dies über ein bestimmtes Interface. In der Regel mit einem Mikro - USB Kabel welches direkt am Computer angesteckt ist. Die Entscheidung fiel auf den Raspberry Pi, aufgrund der geringen Leistung, welche ein Arduino im Vergleich zu einem Raspberry Pi besitzt.

2.2.5 Raspberry Zero

Raspberry Pi Zero ist eine Sparvariante des typischen Raspberry Pi. Er kostet um die 20€. Einsparungen wurden getroffen, wie weniger Ports oder wenn, dann in einer Mini- oder Mikro - Variante. Nur 512 Megabyte an Arbeitsspeicher und nur 1 Kern sind auf dem Raspberry Zero vorhanden. Wir haben uns gegen diese Variante des Raspberry entschieden, da dieser einfach nicht genug Leistung für unsere Anwendung bietet.

2.2.6 Motion

Motion wurde ursprünglich entwickelt um Bewegung vor einer Kamera zu erkennen, daher kommt auch der Name. Bewegung heißt auf Englisch Motion. Motion bietet eine Vielzahl an Konfigurationsmöglichkeiten wie etwa die Schwelle zur Einschätzung, ob eine Bewegung vor der Kamera stattfindet oder nicht. Motion hat einen eingebauten HTTP Server, welcher die aufgenommen Bilder der Kamera mittels Motion JPEG über einem konfigurierbaren Port auf dem Hostsystem zur Verfügung stellt. Konfigurierbar ist bei Motion wirklich alles, von der Kompressionsrate der JPEG bis zu der Pixel genauen Einstellung der Auflösung. Mittels dieser umfangreichen Konfiguration kann Motion, auf jede Situation maßgeschneidert werden, um die Ressourcen jeder Plattform bestmöglich auszunutzen.

2.3 Node JS [SD]

Node.js ist eine Laufzeit Umgebung die auf der Chrome V8 Engine basiert, diese Engine wurde ursprünglich von Google für ihren Browser Chrome entwickelt und wird auch heute noch darin eingesetzt. Die Verwendung der V8 Engine bietet viele Vorteile für Node.js, so ist immer sichergestellt, dass die Software immer auf dem aktuellen Stand ist. Durch die asynchrone Programmierweise eignet es sich vor allem für Webserver oder andere serverseitige Anwendungen. Man kann Node.js wahlweise in Typescript oder Javascript schreiben, wobei angemerkt werden muss das Node.js kein Typescript interpretiert, sondern dieses zu Javascript transpiliert. Für Abhängigkeitsmanagement setzt Node.js auf den npm - Paketmanager.

2.3.1 Vorteile

- Geschwindigkeit

Durch die Verwendung der V8 welche eine der schnellsten Javascript Engines ist, erzielt die Anwendung auch eine hohe Geschwindigkeit.

- Programmiersprache(n)

JavaScript oder Typescript sind Programmiersprachen, die sehr einfach zu lernen sind, also gerade für Anfänger ideal. Hinzu kommt noch, dass man diese Sprache auch für Frontend Frameworks wie Angular einsetzen kann. Man könnte mit dieser Programmiersprache also ein Entwickler sein der an allen Bereichen eines Projektes mitarbeiten kann, dies wird auch Fullstack - Developer genannt.

- Unterstützung

An Beispielen mangelt es nicht, denn mit einer so großen Gemeinschaft wie es Node.js hat findet man sehr viele Demos, Referenz Implementationen oder ganze Projekte auf Github, Gitlab oder anderen Plattformen.

- Asynchrone Programmierung

Ein asynchroner Programmierstil ist, wenn das Framework während es auf eine längere Antwort wartet, nicht einfach blockiert, sondern andere Code-teile ausführt. Dies muss der Programmierer aber explizit mit dem `async` Keyword ausdrücken. Das kann zu einer drastischen Verkürzung der Abfragezeiten führen und dabei können gleichzeitig mehrere Abfragen bearbeiten werden.

- Package Manager

Der Package Manager npm, Abkürzung für Node Package Manager, ist auch sicherlich ein Vorteil, denn er bietet eine sehr große Anzahl von verwendbaren Modulen, welche von anderen Node.js - Entwickler entwickelt und zur Verfügung gestellt werden. Das macht Node.js sehr vielseitig und anpassungsfähig.

- NPM Paket

Die Erstellung eines Node.js Paket ist auch keine Schwierigkeit und somit kann man seine Applikationen abkapseln und in großen Projekt einfach mittels einer Referenz wieder verwendet.

2.3.2 Nachteile

- Updates

Der größte Nachteil von Node.js ist die häufige Aktualisierung der Pakete, wöchentlich wird mindestens 1 Paket upgedatet. Das liegt einfach daran das Node.js noch sehr jung ist. Eine Long Term Solution ist eine Version, welche über längere Zeit nicht großartig verändert wird. Änderungen dürfen auch nicht zur Folge haben, dass der Code umgeschrieben werden muss. Während bei anderen Anbietern LTS jahrelang leben bietet Node.js nur 6 Monat Lebensdauer an. Danach gibt es für diese Version keine Updates mehr und man ist gezwungen, auf eine neuere Version umzusteigen.

- Keine Rückwärtskompatibilität

Nach einem Update kann es auch passieren, dass der Code nicht mehr funktioniert. Der Grund hierfür ist das Node.js keine Rückwärtskompatibilität verspricht. Bei manchen Teilen ist dies zwar der Fall, aber auch nicht immer.

- Architekturproblem

Node.js verwendet nur einen CPU - Thread. Durch ein schlaues Prozesseinteilungssystem können zwar viele kleinere Aktivitäten verarbeitet werden. Bei rechenintensiven Applikationen stößt Node.js schnell an Grenzen, denn es verwendet nicht das ganze Potenzial, welches moderne Computer durch ihre vier, acht oder mehr Kerne bieten könnten.

2.3.3 Alternative: Go

Go ist eine relative neue Sprache, welche Von Google entwickelt und gewartet wird. Entwickelt wurde Sie um Programme effizienter und schneller zu machen. Der Code soll aufgrund der wenigen Schlüsselwörter erzwungenermaßen einfach gehalten werden, dies resultiert in schnelleres lesen und lernen von bereits vorhandenem Code.

2.3.3.1 Vorteile von Go

Go wurde auch speziell für eine Mikroservice Architektur entwickelt. Eine Mikroservice Architektur beschreibt eine Software welche anstatt auf einem einzelnen großen Server, oder auch Monolith genannt, auf viele kleinere setzt. Um dies zu erzielen, läuft Go ohne Laufzeitumgebung, dafür wird Go direkt in Maschinencode compiliert. Ein durchschnittlicher Go Service braucht unter 10 Megabyte an Speicher, im Vergleich dazu alleine die Laufzeitumgebung von Node.js benötigt 60 Megabyte.

2.3.3.2 Nachteile von Go

Die Sprache ist eher jung und noch nicht sehr etabliert. Go kann auch nur auf dem Rechner ausgeführt werden, auf welchem es kompiliert wurde. Durch die minimalistische Anzahl an Schlüsselwörter ist der Einstieg eine Hürde, weshalb man komplett Umdenken muss, dies kostet sehr viel Zeit.

Die Wahl fiel auf Node.js aufgrund bereits erworbener Erfahrung.

2.3.4 Alternative: Asp.Net

Asp.Net ist ein Framework für die Erstellung von serverseitigen Anwendungen. Der Hersteller ist Microsoft und baut auf .Net Core auf .NET Core ist eine Softwareumgebung welche in C# geschrieben wurde und auch C# als ihre Programmiersprache einsetzt. Seit 2016 wurde Asp.Net zu einem Open Source Tool, Microsoft hat hier aber die obere Hand und ist an der Weiterentwicklung maßgeblich beteiligt. Asp.Net setzt auch auf eine asynchrone Programmierung doch diese Funktioniert anders als bei Node.js, während Node.js beim Warten auf Codeteile den Prozess weitergibt, erstellt C# einen neuen Thread der dann anstatt des Hauptprozesses auf diesen wartet und ihn anschließend weiterverarbeitet.

2.3.4.1 Vorteile von Asp.Net

Vorteil von Aps.Net ist sicherlich die Ausgereiftheit der Plattform. Hier werden selten Updates veröffentlicht und der Anwender wird sehr selten gezwungen Code aufgrund eines Versionsupdates zu unternehmen. Weil Microsoft als ein gigantischer Softwarehersteller dahinter steht kann man davon ausgehen das Asp.Net nicht so bald verschwinden wird.

2.3.4.2 Nachteile von Asp.Net

Nachteil von Asp.Net ist sicherlich, dass es erst seit 2016 auf allen Plattformen verfügbar wurde und auf diesen auch noch nicht gänzlich getestet worden ist. Für das Hosten einer Applikation wird eine Windows Server Lizenz fast schon vorausgesetzt, welche man teuer von Microsoft erwerben muss.
Die Entscheidung fiel auf Node.js da diese Applikation einfach zu Dockern ist und die gleiche Sprache wie im Frontend zum Einsatz kommt.

2.4 npm [SD]

Der Node Package Manager ist nicht nur eine gewöhnliche Package Registry auf der Pakete hochgeladen und gespeichert werden können, sondern auch eine Konsoleanwendung. Der Paketmanager weiß, dass er gerade in einem Verzeichnis ist in dem eine Node.js läuft, wenn sich in diesem oder einem übergeordneten Verzeichnis eine package.json Datei befindet. In dieser Datei stehen alle Informationen zum Projekt wie etwa den Autor, Kontakt E-Mail oder eine kurze Beschreibung des Projekts. Alle Abhängigkeiten werden in dieser File entweder händisch eingetragen oder über die Konsole hinzugefügt. Bei Abhängigkeiten gibt es 2 verschiedene Arten, einmal die Entwicklungsabhängigkeiten und die Regulären. Entwicklungsabhängigkeiten werden nur bei dem Weiterentwickeln des Projektes gebraucht, ein bekanntes Beispiel sind Dokumentationen zu Typen, welche die Entwicklungs-Umgebung braucht um wertvolle Informationen und Vervollständigung bieten zu können. Es ist auch möglich Konsolenbefehle in diesem File zu definieren und diese über npm auszuführen.

2.4.1 Alternativen

Yarn wurde 2016 von Facebook veröffentlicht und wurde entwickelt um npm vollständig zu ersetzen. Es bietet also alle Features die auch npm bietet, erweitert diese aber noch um einige Features.

Erstes Feature ist, dass mit yarn die Möglichkeit besteht eine Backup Registry hinzuzufügen, falls Pakete einmal von der Haupt Registry nicht geladen werden können.

Es lädt Pakete auch parallel herunter und speichert diese dann Lokal ab. Wenn dieses Paket das nächste Mal in der gleichen Version benötigt wird, wird es nicht heruntergeladen, sondern von dem Yarn Cache geladen.

Die Entscheidung fiel auf npm da unser Projekt relativ klein ist und so von dem parallelen Herunterladen nicht profitieren würde. Das Sicherheitsrisiko haben wir vernachlässigt aufgrund der bereits erwobenen Erfahrungen mit NPM.

2.5 Angular [SD]

Angular ist ein Open - Source - Framework zur Entwicklung von Single - Page Applikationen auf allen Plattformen wie Smartphones oder Desktop. Es ist zwar ein Open - Source - Projekt, doch Google führt die Entwicklung an und gibt auch maßstäblich die Richtung dafür an. Es wird über den Node Package Manager installiert. Für Angular werden 3 Programmiersprachen benötigt. Zum Schreiben von logischen Codeblöcken wird Typescript verwenden. Für das Layout wird HTML verwendet und CSS wird für die Verschönerung und Anpassung der Benutzeroberfläche verwendet.

2.5.1 Funktion

Angular wird in dieser Diplomarbeit zum Einen für die Ansicht der Messwerte auf einer Grafischen Benutzeroberfläche verwenden, die von jedem Laien verstanden werden kann. Und zum Anderen für die Darstellung und Interaktion mit dem 3d - Model der HTBLA Leonding. Für das 3d - Model wurde aber noch das Three.js Paket verwendet.

2.5.2 Vorteile

- Responsive Design

Angular vereinfacht es dem Entwickler eine Variable aus dem Code anzuzeigen und auch die Ansicht zu ändern, falls sich der Wert dieser Variable ändert. Dies erspart mühsame Arbeit für die Entwickler.

- Abhängigkeiten

Falls an zwei unterschiedlichen Stellen ein gleicher Dienst, wie beispielsweise eine Datenverwaltung benötigt wird, bietet Angular ein Einfügen dieser Abhängigkeiten so dass beide Stellen ein und denselben Service verwenden können.

- Erweiterbarkeit

Die große Community rund um Angular hat bereits etliche Module entwickelt, welche ganz einfach eingebunden werden können. Diese Module können von einer anderen Kommunikationsschnittstelle bis hin zu fertigen grafischen Elementen reichen.

- Model - View - Controller - Pattern

Durch eine Aufgabenteilung zwischen der Logik und der Ansicht wird der Code übersichtlicher und einfacher zu verstehen.

2.5.3 Nachteile

- Steile Lernkurve

Angular Basic sind zwar schnell gelernt, doch es gibt mittlerweile eine Unzahl an Möglichkeiten, welche gerade Anfänger schnell aussteigen lässt.

- Speichergröße

Aufgrund der unzähligen Node Packages welche Angular benötigt kann eine kleine Angular Anwendung schnell mehrere hundert Megabyte groß werden.

- Schneller Releasewechsel

Jedes halbe Jahr kommt eine neue Version von Angular heraus und nach einem Update müssen dann auch alle Module und Pakete aktualisiert werden.

2.5.4 Alternative React

React ist ein von Facebook entwickeltes Open - Source - Framework, welches genau den gleichen Zweck wie Angular erfüllen sollen. Unterschied ist hier das die Ansicht im Code generiert wird und eine Trennung somit nicht mehr vorliegt. Es verwendet auch rxjs was eine reaktive Bibliothek von Javascript ist, anstatt Typescript. Die Entscheidung fiel aufgrund bereits vorhandener Erfahrungen auf Angular.

2.5.5 Angular Material

Angular Material ist eine Implementation aller herkömmlichen Widgets im Material Standard. Der Material Standard ist ein von Google entwickelter Standard zur Vereinheitlichung von Benutzerinterfaces und Widget im Allgemeinen. Dies soll dem Benutzer helfen, sich auf allen verschiedenen Webseiten oder Apps sofort zurechtzufinden. Dies passiert durch vereinheitlichte Symbole und deren Eigenschaften. Es soll auch dem Entwickler Zeit für die Verschönerung ersparen, um dem Anwender später eine intuitive Oberfläche zu bieten.

2.6 Three.js [SD]

Three.js ist eine Javascript 3d Bibliothek welche den standardmäßigen WebGL - Renderer benutzt, dieser läuft auf dem Gerät des Anwenders. Mithilfe von Node.js kann aber auch ein Server das Rendern für den Anwender übernehmen, dies wird Server - Site - Rendering genannt.

WebGL bietet den Vorteil auf Hardware wie die Grafikkarte eines User zugreifen zu können, um Berechnungen zu optimieren.

Es bietet eine Vielzahl an Möglichkeiten angefangen bei einem 3d - Model mit dem der User interagieren kann und sofort ein visuelles Feedback bekommt bis zu 3d - Model welche in Werbungen oder Katalogen eingesetzt werden könnten.

Ein wichtiger Teil von der 3d-Animation ist die dazugehörige Mathematik.

Three.js beinhaltet diese auch direkt im Gegensatz zu Alternativen welche eine weitere Bibliothek brauchen.

2.6.1 Vorteile

- Einfach zu lernen

Selbst, wenn man mit 3d Modellierung nicht allzu viel zu tun hat fällt der Einstieg in three.js leicht.

- Performance

Aufgrund des WebGL Renderers läuft Three.js auf so gut wie allen Plattformen flüssig.

- Dokumentation

Die Dokumentation ist aktuell und sehr detailliert.

2.6.2 Nachteile

- Keine Effekte

Wer nach Effekten wie Partikel, Ton oder physikalischer Logik sucht, wird diese bei Three.js nicht finden.

- Einfache API

Die API wurde gerade für Anfänger oder Fortgeschrittene entwickelt, komplexere Möglichkeiten werden Profis abgehen.

2.7 Postgres [SD]

PostgreSQL oder auch nur Postgres genannt, ist eine Open - Source - Objekt - Relationale Datenbank. Es verwendet das standardmäßige SQL und erweitert diese um Features.

Eine Datenbank kümmert sich um das effiziente Schreiben, Lesen und Speichern der Daten für eine Anwendung.

In manchen Fällen weicht die Syntax vom Standard ab, oder Features werden nicht implementiert, wenn dadurch eine schlechte Architektur gefördert wird.

PostgreSQL ist erweiterbar durch eigene Extensions welche die Datenbank sehr vielseitig macht und dabei hilft, wenn es um die Skalierung einer Applikation geht. Beispielsweise können sogenannte Stored Prozeduren in mehreren Sprachen wie unter anderem Python geschrieben werden. Somit muss der Entwickler sich nicht etwa wie in Oracle die Programmiersprache PlSQL aneignen.

2.8 MQTT [SD]

Message Queuing Telemetry Transport Protokoll ist ein effizientes und ressourcensparendes Protokoll, welches gerade im Internet of Things Bereich sehr beliebt ist. MQTT ist eine “Gerät zu Gerät” - Kommunikation, welche nur eine minimale Bandbreite benötigt. Das Prinzip dahinter ist ein Publish and Subscribe, oder zu deutsch Veröffentlichen und Abonnieren, System. Man teilt die Applikation in Topics auf, Topic können auch untergeordnete Topics besitzen. Sensoren können auf Topics Daten veröffentlichen. Clients können Topics abonnieren, um neue Daten zu erhalten, sobald auf diesen neue Werte gesendet werden. Der Vermittler zwischen den Geräten nennt man MQTT - Broker.

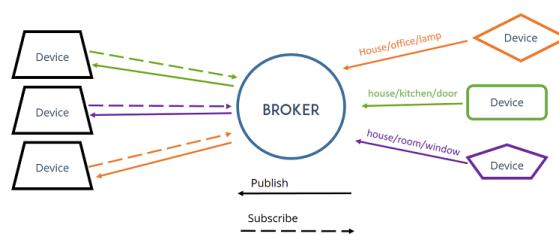


Abbildung 2.3: Mqtt [15]

2.9 Http [SD]

Hypertext Transfer Protocol ist ein Protokoll zur Übertragung von Daten. Es ist ein Server - Client - Protokoll dass heißt, der Server bekommt Anfragen von Client und beantwortet diese, weiters ist es zustandslos, es speichert also keine Daten von Client zwischen Anfragen. Der Hauptanwendungsfall für dieses Protokoll ist im Webbereich. Das Laden einer Website ist nicht anderes als ein HTTP-GET Requests.

2015 wurde HTTP/2 vorgestellt, welches Daten schneller und effizienter übertragen kann. Dies wird durch eine Datenkomprimierung erzielt und ein Aufteilen der einzelnen Komponenten eines Request, wie dem Header und Body, diese werden dann gleichzeitig gesendet. HTTP/2 wird von Browser nur verwendet, wenn die Verbindung HTTPS ist, also sicher verschlüsselt ist.

2.9.1 Https

HTTP Secure ist eine Erweiterung von HTTP mit einer SSL/TLS Schicht. Diese übernimmt die Verschlüsselung und Entschlüsselung von Anfragen und Antworten. Es stellt dabei sicher das Antworten auch wirklich vom abgefragten Server kommen und auch nur dieser die versendeten Daten lesen kann.

2.9.2 Tls Ssl

Transport Layer Security ist eine aktualisierte Version von Secure Socket Layer, der Begriff SSL wird jedoch aufgrund seiner Bekanntheit immer noch eingesetzt. Um eine verschlüsselte Verbindung aufzubauen wird auf ein “Handshake” oder zu Deutsch ein austauschendes Verfahren gesetzt. Dieser läuft wie folgt ab:

- Aufbau

Der Client sendet eine Nachricht an den Server, in der er seine Präferenzen bezüglich des Verschlüsselungsalgorithmus preisgibt und welche Version von SSL verwendet werden sollen.

- **Zertifikat Austausch**

Der Server sendet nun dem Client sein Zertifikat, in diesem stehen wichtige Daten des Server wie der Name des Eigentümers oder etwaige Kontaktinformationen. Ein Zertifikat kann selbst erstellt werden diese nennt man “self-signed” Zertifikate. Um einem Zertifikat wirklich zu glauben, werden Informationen bei vertrauten Zertifikatstellen abgefragt. Bei sehr sensiblen Informationen kann der Server auch ein Zertifikat des Client anfordern.

- **Schlüssel Austausch**

Welche für die Verschlüsselung und Entschlüsselung ein und denselben Schlüssel benützen. Für die Übertragung dieses Schlüssels, welche beide Seiten benötigen, muss ein asynchrone Verschlüsselung verwendet werden. Asynchrone Verschlüsselungen verwenden für die Verschlüsselung einen anderen Schlüssel als zum Entschlüsseln. Für eine solche Verschlüsselung verwendet der Client den öffentlichen Schlüssel der im serverseitigen Zertifikat beinhaltet ist und sendet diesen verschlüsselt, sodass der Server mit seinem privaten Schlüssel, welcher den Server nie verlassen hat, entschlüsseln kann.

Nun können Daten sicher Übertragen werden.

2.10 Rest [SD]

Representational State Transfer ist kein Protokoll oder Standard es ist lediglich eine Architektur die das Kommunizieren zwischen Server und Client vereinfachen soll. Dabei baut es auf Standard wie HTTP bzw. HTTPS auf und sendet Daten meist im JSON-Format, andere Formate wie XML können genauso übertragen werden. JSON hat sich aber als Industriestandard durchgesetzt. Durch die Trennung von Server und Client ist eine Datensammlung im Client oft nur in minimaler Form von Cookies oder ein kurzes Zwischenspeichern von Daten nötig. Das fördert das Portieren von Applikationen auf unterschiedliche Plattformen und erhöht die Skalierbarkeit der Anwendung.

Bei Rest gibt es verschiedene Funktionen die jeweils andere Aufgaben übernehmen sollen. Diese sollen der Leserlichkeit und Dokumentation dienen, müssen aber nicht eingehalten werden.

Die wichtigsten Funktionen darunter sind:

- GET
fordert Daten von einer bestimmten URL des Servers an.
- POST
übermittelt Daten über einen angehängten Body an den Server.
- PUT/PATCH
ändert bestehende Daten am Server
- DELETE
löscht Daten am Server

2.11 Git [SD]

Git ist ein freies Open - Source Versionsverwaltungssystem auf verteilten Systemen. Es eignet sich sowohl für große und kleine Projekte. bleibt dabei sehr performant und effizient. Es ist einfach zu lernen und durch sein Branching Design sticht es aus der Konkurrenz heraus.

Branches sind unabhängige Zweige von der Versionsverwaltung in welchen Entwickler Code austesten können. Durch einen Wechsel des Branches kann ganz einfach an einer anderen Stelle im Versionsbaum weitergearbeitet werden ohne etwaige Änderungen verwerfen zu müssen. Änderungen können einfach gestashed werden, man kann dies mit einem zur Seite legen vergleichen.

Branches bestehen aus mehreren Änderungen sogenannte Commit's. Dieses Design hat den Vorteil Speicher zu sparen und den Vorteil das ein Commit auf einen anderen Branch auch angewandt wird.

Durch die Trennung in Branches kann für jedes Feature ein neuer Branch erstellt werden. Dies hat eine saubere Trennung zur Folge und vereinfacht das Arbeiten an unterschiedlichen Features zugleich.

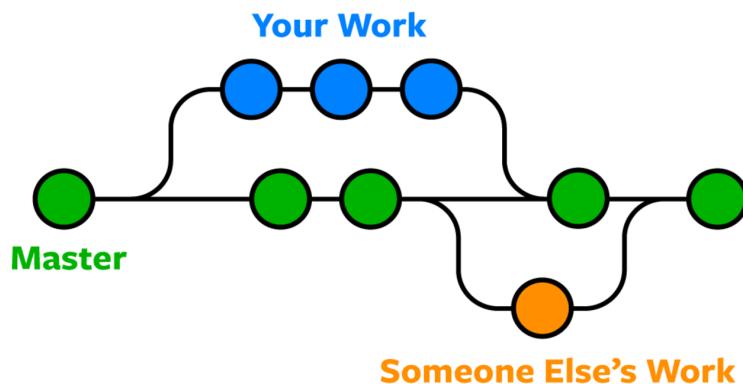


Abbildung 2.4: Git - Visuell [4]

2.12 Docker [SD]

Docker ist eine freie Software zur Containerisierung von Anwendungen. Containerisierung bedeutet, wenn man Anwendung abkapselt um eine Unabhängigkeit von der Plattform und etwaigen installierten Softwarepaketen zu erhalten. Man kann Docker also wie eine eigene virtuelle Maschine sehen, welche aber auf unnötige Features wie etwa eine Grafische Benutzeroberfläche oder Netzwerkadapter gar nicht verwendet oder diese Aufgabe dem Host System überlässt.

Es ermöglicht Entwickler Software, wie Datenbanken schneller verwenden zu können. Wenn bereits Docker installiert ist, kann es in wenigen Sekunden oder Minuten lauffähig sein. Docker basiert auf Linux - Container und war anfänglich nur eine Schnittstelle dafür.

Seit einigen Jahren verwendet Docker eine eigens entwickelte Programmierschnittstelle, welche in Golang entwickelt worden ist. Docker benötigt einen Linux - Kernel um zu funktionieren. Um dies auf Windows und MacOS zu erreichen, wird hier einfach eine kleine virtuelle Box mit dem Linux - Kernel gestartet.

2.12.1 Grundbegriffe

- **Image**

Ist eine Ansammlung von Layern welche auf das Basis-Image gestapelt werden. Das Basis-Image ist ein sehr kleines Image, welches nur eine binär-Datei ausführen kann.

- **Container**

Ein Container ist eine Instanz eines Images, dieses kann verschiedene Ports verwenden und Daten speichern. Container können gestartet und gestoppt werden.

- **Volume**

Container haben einen Speicher welcher an den Container gebunden ist. Wird der Container gelöscht gehen die Daten verloren, man kann jedoch Dateien nach außen binden, um diese zu sichern.

- **Layer**

Ein Layer ist ein Teil des Images welcher Software zum Image hinzufügt oder Files. Layer haben den Vorteil ,dass wenn eine Software in 2 unterschiedlichen

Images verwendet wird, jede Layer nur 1 mal heruntergeladen und gespeichert werden muss. Es sind also Bauklötze die aufeinander gestapelt werden, um ein Image zu erhalten.

- **Dockerfile**

Das Dockerfile ist ein einfaches Textfile in dem man die Dockerbefehle niederschreiben kann. Man kann in diesem das Basisimage angeben, Zwischenschritte definieren oder Multi - Stage - Builds ausführen. Pro Zwischenschritt wird eine Layer generiert und gespeichert.

- **Multi - Stage - Build**

Man spricht von Multi-Stage build wenn man mehrere verschiedene Images verwendet, um das gewünschte Image zu erhalten. Hierbei kann eine Stage den Teil des Bauens der Applikation übernehmen und ein anderes das Ausführen. Vorteil ist das der resultierende Container kleiner ist, da er nicht mehr alle Tools, welche zum Bauen der Applikation verwendet werden beinhaltet, sondern nur mehr jene die zum Ausführen benötigt werden.

- **Registry** Registry ist ein Speicherort für Images, diese beinhalten auch die Versionsverwaltung, wobei die bekannteste davon Dockerhub ist.

2.12.2 Docker-compose

Docker-compose ist eine Technologie, mit der das Verwalten mehrerer Container auf einem Rechner vereinfacht wird. Wichtig ist, dass es keine Alternative zu Container Orchestration Tool wie Kubernetes ist, es soll nur das Starten beziehungsweise das Stoppen mehrerer Container übernehmen. Durch das einfache Verlinken von verschiedenen Containern wird dies oftmals eingesetzt, wenn eine Kommunikation unter den Containern benötigt wird.

2.12.3 Traefik

Traefik ist ein Reverse Proxy, dieser übernimmt die Aufgabe einzelne Routen von einem Server auf die einzelnen Applikationen zu verlagern. Nach außen muss dann nurmehr 1 Port für Traefik freigegeben werden. In der Regel ist dies der Standardport für HTTP: 80 oder der Standardport für HTTPS 443. Traefik unterstützt auch die Verschlüsselung über HTTPS. Loadbalancing wird zwar von Traefik angeboten doch dieses Feature haben wir nicht verwendet.

2.13 Künstliche Intelligenz - Machine Learning [CP]

Künstliche Intelligenz kann sehr generell definiert werden, als die Fähigkeit von Computersystemen oder Computer gesteuerten Robotern Aufgaben zu erfüllen, die typischerweise intelligente Wesen, etwa Personen oder Tiere, zur Erfüllung verlangen.

Bei einer solch breiten Auslegung der Definition wird jedoch nicht klargestellt, wie es zu der nötigen Intelligenz kommt. Das gesamte intelligente Verhalten könnte beispielsweise aus den Erkenntnissen des Entwicklers stammen, welcher diese zu strikten Abläufen in der Form eines Programms übersetzt.

Durch die Fortschritte und Innovationen im Feld Machine Learning, einem Unter teil von Künstlicher Intelligenz, hat sich die Definition von Künstlicher Intelligenz immer mehr der von Machine Learning genähert.

Arthur Samuel, ein Pionier im Feld Künstliche Intelligenz, popularisierte den Begriff "Machine Learning" im Jahre 1959 und beschrieb dieses Feld informell als "The field of study that gives computers the ability to learn without being explicitly programmed"[11].

Eine modernere Definition wird von Tom Mitchell, einem Forscher bekannt für seine Beiträge zur Förderung von Machine Learning, gegeben: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [6]

In unserem Zeitalter, in dem jedes Gerät, jede Maschine, Sensoren enthält und jeder Ablauf, jede Aktion, genauestens dokumentiert wird, sind beides, strukturierte und unstrukturierte Daten, im Überfluss vorhanden. Anstatt Menschen zu benötigen, die regelbasierte Programme bauen, bietet Machine Learning eine effizientere Alternative Daten zu verarbeiten und ermöglicht dadurch das Treffen datengetriebener Entscheidungen.

Die Auswirkungen des rapiden Fortschritts in Machine Learning sind jedoch nicht nur für große Unternehmen zu spüren, auch im Alltag bringt es einige Änderungen. Dank Machine Learning gibt es robuste E-Mail Spam Filter, Sprach- und Texterkennung, verlässliche Suchmaschinen und vieles mehr.

2.13.1 Mathematische Notation

Um die folgenden mathematischen Ausdrücke so klar wie möglich zu machen wird hier die Notation für Benennung von Variablen und deren mögliche Zugriffsarten erklärt.

2.13.1.1 Skalar

Ein Skalar ist ein einfacher numerischer Wert wie 7 oder -5.29. Variablen oder Konstanten mit skalaren Werten werden kursiv geschrieben, wie etwa x oder a , bezeichnet.

2.13.1.2 Vektor

Ein Vektor ist eine geordnete Liste von skalaren Werten, genannt Attribute. Vektoren werden mit fett gedruckten Buchstaben bezeichnet, wie \mathbf{x} oder \mathbf{w} . Ein Attribut eines Vektors wird als kursiver Buchstabe mit einem Index angegeben, wie $w^{(j)}$ oder $x^{(j)}$. Der Index j ist eine spezifische Dimension des Vektors, die Position eines Attributes in der Liste. Hier ein Beispiel:

Der Wert des Vektors \mathbf{a} wird festgelegt mit

$$\mathbf{a} = [-2, 5, 8, -1]$$

und danach werden die Dimensionen 1 und 3 ausgelesen

$$a^{(1)} = -2$$

$$a^{(3)} = 8$$

Eine Variable kann auch 2 oder mehr Indizes haben, wie $x_i^{(j)}$ und $x_{i,j}^{(k)}$. Beispielsweise bezeichnet $x_{l,u}^{(j)}$ das Merkmal j des Neuronen u im Layer l eines Neuronalen Netzwerks.

2.13.1.3 Matrix

Eine Matrix ist ein rechteckiges Array von Nummern, angeordnet nach Zeilen und Spalten. Folgend ist ein Beispiel für eine Matrix mit zwei Reihen und drei Spalten,

$$\begin{bmatrix} -6 & 21 & 2 \\ 4 & -3 & -1 \end{bmatrix}$$

Matrizen werden mit fett gedruckten Großbuchstaben gekennzeichnet, wie **A** und **W**.

2.13.1.4 Funktionen

Eine Funktion verlangt bestimmte Parameter, etwa Skalare und Vektoren, und liefert für diese einen Rückgabewert. Ein Beispiel für eine Funktion ist:

$$f_{\mathbf{w}, b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$$

Der Name der Funktion verwendet die Notation des Rückgabewertes, in diesem Fall wird ein Skalar zurückgegeben, daher ist der Name kursiv. Die Notation $f_{\mathbf{w}, b}(\mathbf{x})$ bedeutet, dass f von \mathbf{w} , b und \mathbf{x} parametrisiert ist. Die Parameter \mathbf{w} und b ändern sich dabei nur beim Training des Modells, wie später genauer erklärt wird, daher werden sie anders als \mathbf{x} angezeichnet.

2.13.2 Arten

Machine Learning kann in vier Unterabschnitte unterteilt werden:

- Supervised Learning
- Unsupervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

2.13.2.1 Supervised Learning

Bei Supervised Learning ist der Datensatz eine Ansammlung von beschrifteten Beispielen $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Jedes Element \mathbf{x}_i , $i = 1, \dots, N$ ist ein so genannter Merkmalsvektor. Ein Merkmalsvektor ist ein Vektor, in dem jedes Attribut, genannt Merkmal, $\mathbf{x}_i^{(j)}$, $j = 1, \dots, D$ das Beispiel beschreibt. y_i wird Label genannt und gibt den richtigen Output für den Input \mathbf{x}_i .

Zur Illustration, falls jedes Beispiel \mathbf{x} in einer Kollektion ein Haus beschreibt, dann könnte das erste Merkmal, $x^{(1)}$, die Anzahl an Räumen sein, das zweite Merkmal, $x^{(2)}$, das Alter des Hauses und so weiter.

Alle Beispiele im Datensatz müssen das gleiche, konsistente Schema benutzen. Das bedeutet, wenn $x_i^{(1)}$ für einen beliebigen Merkmalsvektor \mathbf{x}_i das Alter eines Hauses in Jahren angibt, dann muss $x_k^{(1)}$ für alle anderen Beispiele $\mathbf{x}_k, k = 1, \dots, N$ auch das Alter eines Hauses in Jahren angeben.

Das Label y_i kann entweder einem Set von Klassen $\{1, 2, \dots, C\}$ angehören, dann wird die Aufgabenstellung "Klassifizierung" genannt, oder eine reale Nummer beziehungsweise eine Struktur, wie etwa ein Vektor oder eine Matrix, sein, dann wird die Aufgabenstellung "Regression" genannt.

Ein klassisches Beispiel für Klassifikation ist das eines Spam Filters für E-Mails, bei dem es zwei Klassen gibt $\{\text{spam}, \text{nicht_spam}\}$.

Das Ziel bei Supervised Learning Problemen ist es, ein Modell zu generieren, welches für einen Merkmalsvektor \mathbf{x} als Input das Label y ausgibt. Beispielsweise könnte ein Modell, welches für einen Datensatz von Häusern, wie vorhin besprochen, erzeugt wurde, als Input die Daten eines Hauses bekommen und als Output den Preis des Hauses geben.

2.13.2.2 Unsupervised Learning

Bei Unsupervised Learning ist der Datensatz eine Ansammlung von Merkmalsvektor $\{\mathbf{x}_i\}_{i=1}^N$ ohne gegebene Labels. Das Ziel von Unsupervised Learning ist es ein Modell zu erschaffen, welches einen Merkmalsvektor \mathbf{x} zu einer Struktur oder einem skalaren Wert transformiert und damit ein gegebenes Problem löst.

Die drei häufigsten Anwendungen für Unsupervised Learning sind:

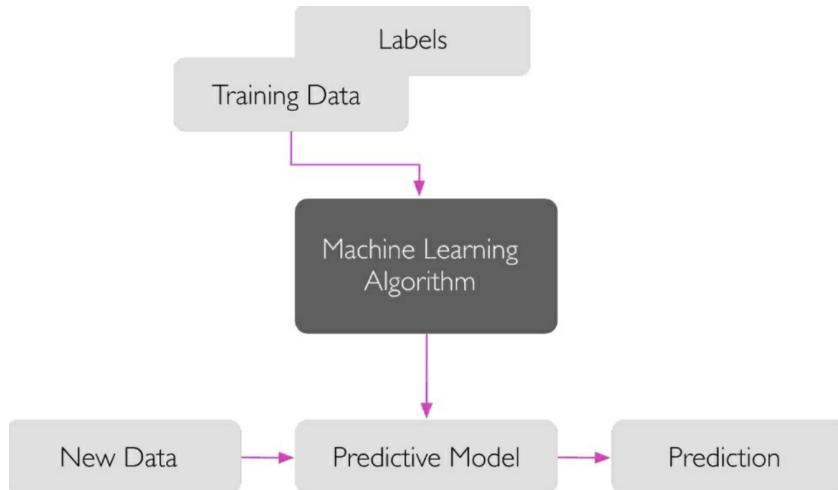


Abbildung 2.5: Illustration des Ablaufes für Supervised Learning.[9]

2.13.2.2.1 Clustering Hierbei gibt das erstellte Modell für einen gegebenen Merkmalsvektor \mathbf{x} die ID eines Clusters zurück. Dies scheint sehr ähnlich zu Supervised Learning mit Klassen als Output, Klassifizierung, jedoch ist es die Aufgabe des Lernalgorithmus diese Klassen zu finden, da sie ja nicht gegeben sind.

2.13.2.2.2 Dimensionsreduktion Das Ziel ist es, für einen gegebenen Merkmalsvektor \mathbf{x} einen Vektor mit geringerer Dimensionalität auszugeben, welcher möglichst viele Informationen des anfänglichen Vektors beinhaltet.

2.13.2.2.3 Außenseitererkennung Hierbei gibt das gelernte Modell zurück, wie ähnlich der gegebene Merkmalsvektor allen anderen Merkmalsvektoren im Datensatz ist. Dies kann beispielsweise mit Netzwerkzugriffen trainiert werden, um auffälliges Verhalten im Netzwerk zu entdecken.

2.13.2.3 Semi-Supervised Learning

Bei Semi-Supervised Learning enthält der Datensatz sowohl Beispiele mit als auch ohne Labels. Üblicherweise enthält der Datensatz um einiges mehr Beispiele ohne Labels als mit. Die Hoffnung hierbei ist, dass diese Menge an Beispielen ohne Labels dem Lernalgorithmus helfen kann, ein besseres Modell zu erzeugen. Sie geben dabei zusätzliche Einsicht in etwa die Verteilung der Daten, welche der Lernalgorithmus nutzen können sollte.

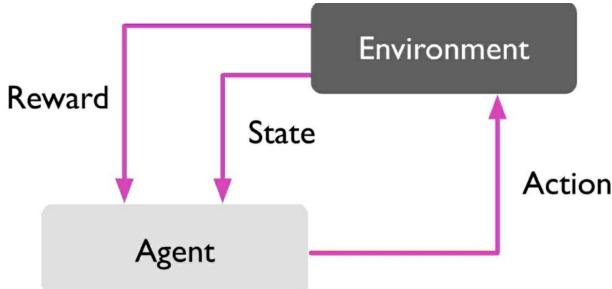


Abbildung 2.6: Illustration des Ablaufes für Reinforcement Learning.[9]

2.13.2.4 Reinforcement Learning

Bei Reinforcement Learning "lebt" die Maschine in einer Umgebung. Sie kann den Zustand in dem sich die Umgebung befindet als Merkmalsvektor auslesen und Aktionen tätigen um mit dem Umfeld zu interagieren. Eine Aktion bringt eine bestimmte Belohnung und kann den Zustand der Umgebung verändern.

Nun soll gelernt werden, für einen bestimmten Zustand der Umgebung die Aktion zu wählen, welche die durchschnittlichen erwarteten Belohnungen maximiert. Dabei wird nicht nur auf die direkte Belohnung optimiert, sondern auch auf Zukünftige.

Reinforcement Learning wird genutzt für Problemstellungen bei denen Entscheidungen sequentiell getroffen werden und das Ziel langfristig ist. Beispiele für Anwendungsfelder sind das Spielen von Spielen, Robotik und Logistik.

2.13.3 Lineare Regression

Lineare Regression ist einer der fundamentalsten Lernalgorithmen für Supervised Learning Problemstellungen. Sie lernt ein Modell, welches eine lineare Kombination aus dem Input ist.

Wie bei Supervised Learning gefordert, ist der Datensatz zum Trainieren des Modells in der Form $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ gegeben. Hierbei ist \mathbf{x}_i der D -dimensionale Merkmalsvektor für Beispiel $i = 1, \dots, N$, y_i ist ein Skalar als Label und alle Merkmale $\mathbf{x}_i^{(j)}, j = 1, \dots, D$ sind auch skalare Werte.

Das Ziel ist es, eine lineare Kombination der Merkmale im Merkmalsvektor \mathbf{x} als

Modell zu kreieren:

$$f_{\mathbf{w}, b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b \quad (2.1)$$

wobei \mathbf{w} ein D -dimensionaler Vektor und b ein Skalar ist. Diese Funktion wird generell auch Hypothesenfunktion genannt.

Da die Hypothesenfunktion optimiert werden soll um bessere Vorhersagen zu treffen und da diese Funktion vom Paar (\mathbf{w}, b) parametrisiert wird, ist es das Ziel, die Wahl dieses Paares zu optimieren.

Die Hypothesenfunktion erzeugt eine Hyperebene. Diese wird bei Linearer Regression so gewählt, dass sie möglichst nahe an den gegebenen Beispielen liegt. Dies ist in Abbildung 2.7 dargestellt. Sie zeigt den Datensatz für einen eindimensionalen Fall verteilt nach ihren x und y Komponenten. Die strichlierte Linie zeigt das gelernte Modell, die Hypothesenfunktion, und wird für Vorhersagen bei neuen Beispielen genutzt. Wenn der Merkmalsvektor \mathbf{x} mehr-dimensional ist wird die Gerade zu einer Ebene ($D = 2$) oder einer Hyperebene ($D > 2$).

Damit die Hyperebene so nahe an den Beispielen liegt wie möglich, ermittelt der Optimierungsvorgang die optimalen Werte für \mathbf{w} und b indem er den folgenden Ausdruck minimiert:

$$\frac{1}{N} \sum_{i=1}^N (f_{\mathbf{w}, b}(\mathbf{x}_i) - y_i)^2 \quad (2.2)$$

Der Ausdruck $(f_{\mathbf{w}, b}(\mathbf{x}_i) - y_i)^2$ wird Loss Funktion genannt und ist eine Messung des Errors beim Vorhersagen für \mathbf{x}_i . Die hier gewählte Loss Funktion wird "squared error loss" genannt. Jeder Lernalgorithmus, der auf einer Hypothesenfunktion basiert, hat eine Loss Funktion und um das beste Modell zu finden wird die Kostenfunktion minimiert. Für Lineare Regression ist die Kostenfunktion der durchschnittliche Loss, in diesem Fall auch "mean squared error" oder MSE genannt, und ist im Ausdruck 2.2 gegeben. Um die Ableitung der Kostenfunktion, welche folgend zur Optimierung der Parameter genutzt wird, etwas zu vereinfachen wird jedoch diese Form genutzt, in der die Summe noch durch 2 geteilt wird:

$$\frac{1}{2N} \sum_{i=1}^N (f_{\mathbf{w}, b}(\mathbf{x}_i) - y_i)^2 \quad (2.3)$$

Ein Vorteil von linearen Modellen ist, dass sie selten overfitten. Wenn ein komplexes Modell sich beim Trainieren zu stark an die gegebenen Beispiele anschmiegt, um die Kostenfunktion zu minimieren, kann dies zu schlechten Vorhersagen bei ungesehenen Beispielen führen. Dieses Verhalten wird Overfitting genannt.

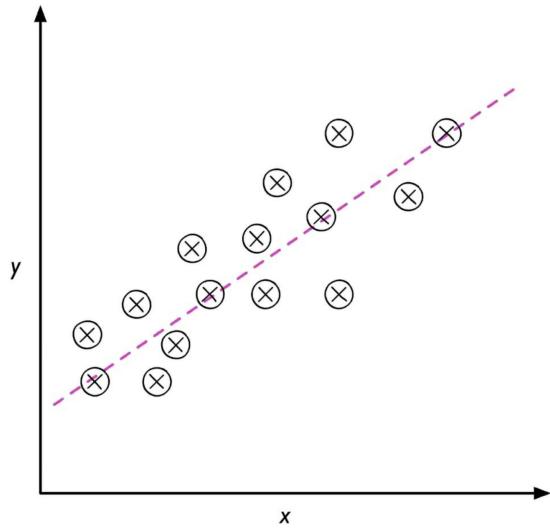


Abbildung 2.7: Lineare Regression für ein-dimensionale Beispiele.[9]

2.13.3.1 Gradient Descent

Gradient Descent ist ein iterativer Ansatz, um lokale Minima von ableitbaren Funktionen zu finden. Bei jeder Iteration, auch Epoche genannt, werden die partiellen Ableitungen für die derzeitigen Parameter ermittelt und ein Schritt in Richtung des steilsten Gefälles gemacht. Die Länge dieses Schrittes ist proportional zur Steigung. Wenn die Steigung 0 ist, also das lokale Minimum erreicht wurde, bewegt sich Gradient Descent daher nicht weiter. Dies wird Konvergenz genannt.

In der Abbildung 2.8 wird die Kostenfunktion für ein ein-dimensionales Beispiel von Linearer Regression gezeigt. Dabei verlaufen die Parameter (w, b) entlang der X- und Y-Achse und die Kosten für diese Kombination von Parametern wird auf der Z-Achse dargestellt. Für Lineare Regression ist diese Abbildung immer konvex und es wird daher immer das globale Minimum gefunden, dies ist jedoch keine Voraussetzung für Gradient Descent. Der Verlauf von Gradient Descent wird hier mit pinken Sternen markiert, jeder Stern zeigt die Position in einer Epoche.

Die Geschwindigkeit des Abstiegs zum lokalen Minimum ist gesteuert durch einen Parameter a , welcher mit der Steigung multipliziert wird, um die Schrittgröße zu ermitteln. Daher je größer a desto schneller konvergiert Gradient Descent, jedoch nur bis zu einer Grenze, danach kommt es zu keiner Konvergenz.

Der Algorithmus für Gradient Descent lautet:

Bis Konvergenz wiederholen: {

$$\theta_j := \theta_j - a \frac{\partial}{\partial \theta_j} J(\theta) \quad (2.4)$$

}

Wobei θ ein Vektor mit allen Parametern ist, also sowohl alle Attribute aus \mathbf{w} beinhaltet als auch b , J die Kostenfunktion ist und a die Lernrate ist. In jeder Epoche wird für jeden Parameter des Modells die Partielle Ableitung ermittelt und ein Schritt in die gegengesetzte Richtung gemacht mit einer Rate von a . Alle Änderungen der Parameter müssen am Ende der Epoche durchgeführt werden, nicht nacheinander.

Für die Kostenfunktion als gegeben im Ausdruck 2.3 mit ein-dimensionalen Beispielen wäre der Ablauf folgender:

Bis Konvergenz wiederholen: {

$$b := b - a \frac{1}{N} \sum_{i=1}^N f_{w,b}(x_i) - y_i$$

$$w := w - a \frac{1}{N} \sum_{i=1}^N (f_{w,b}(x_i) - y_i) x_i$$

}

Gradient Descent ist ein sehr fundamentaler Algorithmus für die Optimierung von Parametern in einem Modell. Es gibt eine Vielzahl von weiteren Optimierern, die in der Praxis Vorteile über Gradient Descent bieten. Diese implementieren Verbesserungen wie etwa das Aufteilen des Datensatzes vor der Berechnung der Ableitung, "batching", oder das dynamische Ändern der Lernrate.

Eine Möglichkeit Gradient Descent zu beschleunigen ist es alle Merkmale in etwa die gleiche Reichweite an Werten zu bringen. Das Ziel ist es $-1 \leq \mathbf{x}^{(j)} \leq 1$ oder $-0.5 \leq \mathbf{x}^{(j)} \leq 0.5$ zu erreichen. Es muss jedoch nicht exakt sein, es geht nur um etwas Beschleunigung. Eine Art dies zu erreichen ist die Kombination von "feature scaling" und "mean normalization", die diese Formel ergibt:

$$\mathbf{x}^{(j)} := \frac{\mathbf{x}^{(j)} - \mu_j}{s_j} \quad (2.5)$$

wobei μ_j der durchschnittliche Wert für Merkmal j ist und s_j die Standard Abweichung von Merkmal j ist.

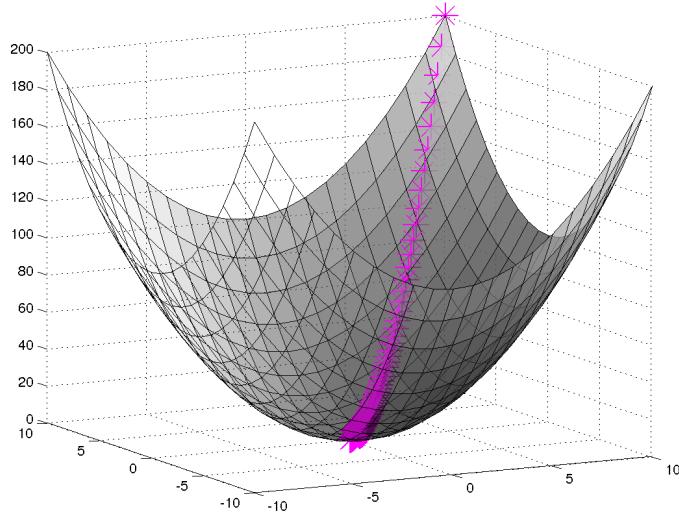


Abbildung 2.8: Darstellung der Kostenfunktion bei ein-dimensionaler Linearer Regression und dem Verlauf von Gradient Descent.[5]

2.13.4 Logistische Regression

Der wichtigste Fakt über Logistische Regression ist, dass es sich hierbei nicht um Regression sondern Klassifikation handelt. Es wird so genannt da der mathematische Hintergrund sehr dem von Linearer Regression ähnelt.

In Logistischer Regression ist $y_i, i = 1, \dots, N$ ein binärer Wert, entweder 0 oder 1. Da bei Linearer Regression der Output von negativ Unendlich bis positiv Unendlich reicht, aber der gewollte Output binär ist, ist es leicht zu erkennen, dass es nicht direkt anwendbar ist und Änderungen gemacht werden müssen.

Die erste Änderung ist das Einführen einer Aktivierungsfunktion, der Standard Logistischen Funktion oder auch Sigmoid Funktion genannt. Diese hat die Eigenschaft, dass sie einen beliebigen Input Skalar umwandelt zu einem Wert im Intervall $(0, 1)$. Sie ist definiert als:

$$a(x) = \frac{1}{1 + e^{-x}} \quad (2.6)$$

Diese Aktivierungsfunktion wird auf das Ergebnis der Hypothesenfunktion aus Linearer Regression angewendet und gibt damit die Hypothesenfunktion für Logistische Regression:

$$f_{\mathbf{w}, b}(\mathbf{x}) := a(\mathbf{w}\mathbf{x} + b) \quad (2.7)$$

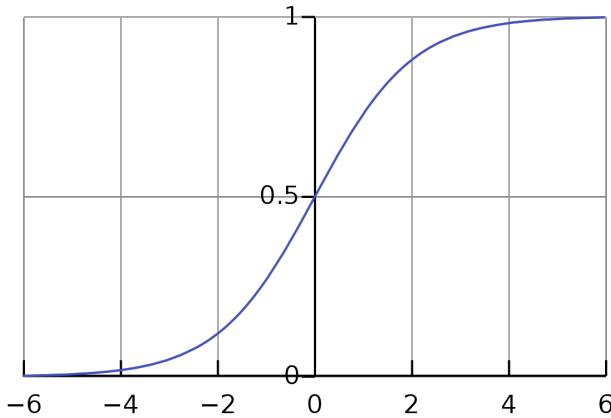


Abbildung 2.9: Darstellung der Sigmoid Funktion.[12]

oder auch als einzelne Funktion:

$$f_{\mathbf{w}, b}(\mathbf{x}) := \frac{1}{1 + e^{-(\mathbf{w}\mathbf{x} + b)}} \quad (2.8)$$

Das Ergebnis dieser Funktion kann angesehen werden als die Wahrscheinlichkeit, dass 1 das Label für \mathbf{x} ist. Das heißt, es kann ein Schnittpunkt festgelegt werden, beispielsweise 0.6, ab welchem 1 als Ergebnis zurückgegeben wird, ansonsten 0. Dieser Schnittpunkt ist problemspezifisch.

Als Beispiel, bei (\mathbf{x}_i, y_i) gibt die Funktion $f_{\mathbf{w}, b}$ für den Input \mathbf{x}_i einen Output p im Intervall $(0, 1)$. Wenn $y_i = 1$, ist die Wahrscheinlichkeit dafür laut dem Modell gegeben durch p . Wenn $y_i = 0$, dann ist die Wahrscheinlichkeit dafür laut dem Modell $1 - p$.

Bei Logistischer Regression wird nicht der MSE wie bei Linearer Regression minimiert, sondern die Wahrscheinlichkeit unseres Datensatzes anhand des Modells maximiert

Dieses Optimierungsziel wird "maximum likelihood" genannt und ist definiert als:

$$J_{\mathbf{w}, b} := \prod_{i=1}^N f_{\mathbf{w}, b}(\mathbf{x}_i)^{y_i} (1 - f_{\mathbf{w}, b}(\mathbf{x}_i))^{1-y_i} \quad (2.9)$$

Der Ausdruck $f_{\mathbf{w}, b}(\mathbf{x}_i)^{y_i} (1 - f_{\mathbf{w}, b}(\mathbf{x}_i))^{1-y_i}$ ist die mathematische Form der Regeln für Wahrscheinlichkeiten aus dem Modell Output wie vor kurzem erklärt.

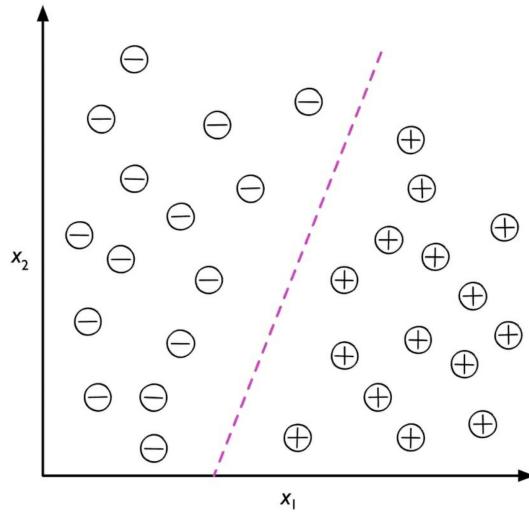


Abbildung 2.10: Darstellung der Trennung von Klassen durch ein gelerntes Modell (strichlierte Linie).[9]

In der Praxis ist es jedoch praktischer die "log-likelihood" zu maximieren, diese ist folgenderweise definiert:

$$\text{Log } J_{\mathbf{w}, b} := \ln (J_{\mathbf{w}, b}) = \sum_{i=1}^N [y_i \ln f_{\mathbf{w}, b}(\mathbf{x}_i) + (1 - y_i) \ln (1 - f_{\mathbf{w}, b}(\mathbf{x}_i))] \quad (2.10)$$

Der Vorteil dieser Form ist, dass Multiplikationen von Werten im Intervall $(0, 1)$ vermieden werden, welche zu Ungenauigkeit des Ergebnisses auf Computersystemen führen kann.

Weiters, um Gradient Descent nutzen zu können welcher Minima sucht, wird das negative der Funktion genommen. Dadurch werden Maxima zu Minima und umgekehrt. Diese endgültige Funktion ist als "negative log-likelihood" bekannt.

2.13.5 Neuronale Netzwerke

Neuronale Netzwerke sind ein Versuch, die Abläufe im Gehirn abzubilden und dadurch sowohl mehr über das Gehirn zu lernen als auch Maschinen intelligenter zu machen.

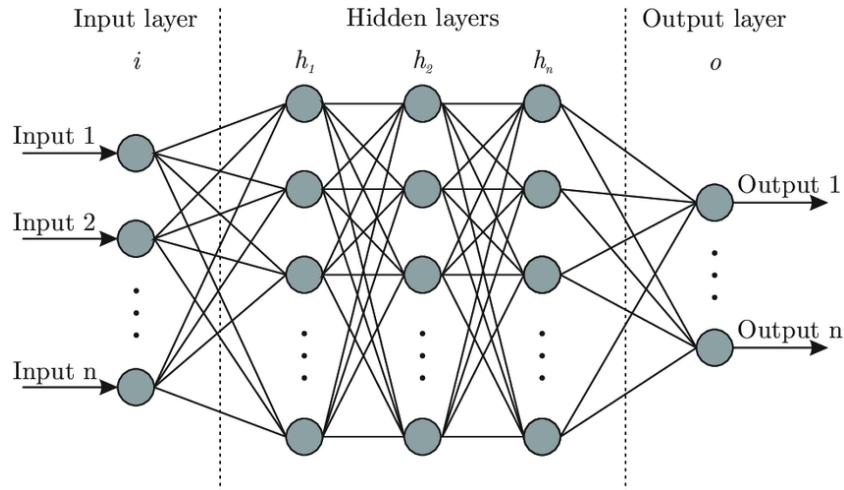


Abbildung 2.11: Illustration eines Neuronalen Netzwerkes.[2]

Ein Neuronales Netzwerk ist aus einzelnen Neuronen aufgebaut, welche im simpelsten Fall wie eine Instanz von Linearer Regression sind, wobei sie oft eine Aktivierungsfunktion wie bei Logistischer Regression beinhalten. Diese Neuronen sind meist in so genannten Layers gruppiert und der Output eines Layers wird zum Input Merkmalsvektor des nächsten Layers. Der Input für den ersten Layer ist der anfängliche Merkmalsvektor \mathbf{x} . Dies Alles ist in der Abbildung 2.11 gezeigt.

Alle Layers zwischen dem Input- und dem Output-Layer werden Hidden-Layers genannt. Beim extrem bekannten "Deep Learning" beinhaltet das Netzwerk einfach eine größere Anzahl an Hidden-Layers.

Für Klassifikation, wenn nur ein Neuron im letzten Layer vorhanden ist, wird die Sigmoid Funktion als Aktivierungsfunktion genutzt, ansonsten wird die Softmax Funktion genutzt. Die Softmax Funktion ist der Sigmoid Funktion sehr ähnlich und hat die zusätzliche Eigenschaft, dass $\sum_{i=1}^M o^{(i)} = 1$, wobei \mathbf{o} der M -dimensionale Output Vektor des letzten Layers ist. Das heißt die Summe aller Wahrscheinlichkeiten ergibt 1, was praktisch auch Sinn macht, da die Wahrscheinlichkeit, dass ein Beispiel \mathbf{x} zu einer der möglichen Klassen gehört, immer 100% sein sollte.

Sie bieten den Vorteil von nicht linearen Hypothesen für komplexere Problemstellungen, kommen jedoch auch mit Nachteilen. Ein Nachteil ist die Gefahr von Overfitting, als bereits bei Linearer Regression erwähnt. Ein weiterer Nachteil ist, dass die Anzahl an lernbaren Parametern rasch mit der Anzahl von Neuronen steigt, was zu langsamem Training führt.

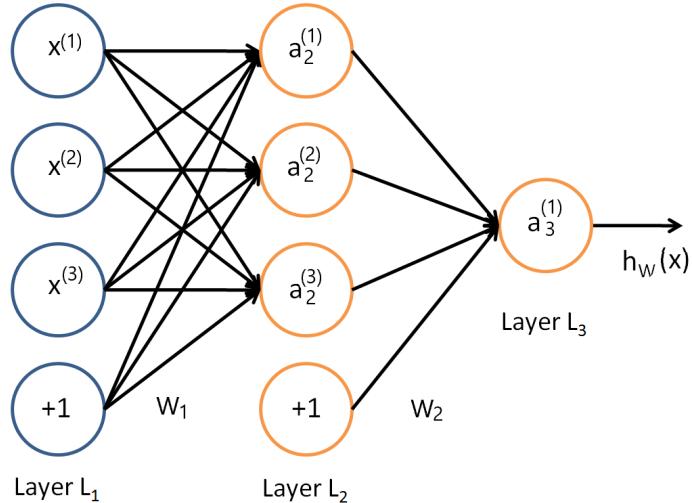


Abbildung 2.12: Illustration von Forward Feeding.[10]

2.13.5.1 Forward Feeding

Forward Feeding beschreibt den Weg, in dem Daten durch das Neuronale Netzwerk fließen.

Bisher wurde die lineare Kombination der Merkmale immer mit $\mathbf{w}\mathbf{x} + b$ berechnet. Eine andere, in vielen Fällen einfacher benutzbare Form davon ist $\mathbf{w}\mathbf{x}$, wobei \mathbf{w} ein Attribut mehr hat als Merkmale vorhanden sind und eine 1 am Anfang des Merkmalsvektors \mathbf{x} hinzugefügt wurde. Praktisch wurde b als extra Attribut zu \mathbf{w} hinzugefügt und mit einer Konstanten 1 im Merkmalsvektor multipliziert, wodurch es nicht vom Input abhängt. Dieser Ablauf ist in der Abbildung 2.12 illustriert.

Die Daten verlaufen durch das Neuronale Netzwerk wie folgt:

$$\mathbf{a}_l = g_l (\mathbf{W}_{l-1} \mathbf{a}_{l-1}) \quad (2.11)$$

wobei \mathbf{a}_l der Output Vektor, gegeben als eine $\mathbb{R}^{n \times 1}$ Matrix, für Layer l mit $l = 1, \dots, L$ ist, \mathbf{W}_l eine $\mathbb{R}^{s_l \times (s_{l-1}+1)}$ Matrix ist, mit s_l als die Anzahl an Neuronen im Layer l , welche alle Gewichtete Vektoren \mathbf{w} für die Neuronen im Layer l beinhaltet ($s_{l-1} + 1$ wegen der extra konstanten 1 für das Bias) und g_l die Aktivierungsfunktion für Layer l ist. Außerdem gilt $\mathbf{a}_1 = \mathbf{x}$. Mit diesem Ablauf kann der Output des Neuronale Netzwerkes ermittelt werden, der Output des Modells ist gleich der Output des letzten Layers.

2.13.5.2 Back Propagation

Die partiellen Ableitungen aller Parameter im Neuronalen Netzwerk in jeder Epoche von Gradient Descent symbolisch oder numerisch abzuleiten ist in der Praxis zu langsam und/oder speicherintensiv. Eine Lösung für dieses Problem ist Back Propagation. Dabei wird das Modell von Hinten nach vorne, Layer für Layer, durchgegangen und alle partiellen Ableitungen für den Layer berechnet anhand dem Error im nächsten Layer und dem Input für den derzeitigen. Daher kommt auch der Name "Back Propagation", der Error in einem Layer wird zum vorherigen weiter propagiert.

2.13.5.3 Aktivierungsfunktionen

Um nicht triviale Probleme effektiv zu lösen benötigen Neuronale Netzwerke die nicht Linearität von Aktivierungsfunktionen. Um Gradient Descent und ähnliche Algorithmen, die die Steigung zum Finden von Minima benutzen, zu ermöglichen, muss die Aktivierungsfunktionen kontinuierlich ableitbar sein (dies ist bei ReLU, gegeben in der Abbildung 2.13, an der Stelle $x = 0$ nicht der Fall, jedoch kann in der Praxis frei 0 oder 1 als Steigung gewählt werden, wodurch es nutzbar wird).

In der Abbildung 2.13 sind einige der wichtigsten Aktivierungsfunktionen für einzelne Neuronen illustriert. Dazu kommt noch Softmax, welches nur auf mehrere Neuronen gleichzeitig angewendet werden kann. Wie vorhin schon erwähnt ähnelt es der Sigmoid Funktion, hat aber die zusätzliche Eigenschaft, dass die Summe der Outputs für die einzelnen Neuronen 1 ergibt.

2.13.5.4 Recurrent Neural Networks

Recurrent Neural Networks sind eine Art von Neuronalen Netzwerken, die sich dadurch auszeichnen, dass sie Schleifen im Netzwerk enthalten. Sie werden dafür genutzt, um Vorhersagen auf temporalen Input Daten zu treffen. Der Input ist dabei als eine Matrix von Merkmalsvektoren gegeben, jeder Merkmalsvektor repräsentiert die Messwerte an einem bestimmten Zeitpunkt und die Positionen der Merkmalsvektoren in der Matrix sind zeitlich sortiert.

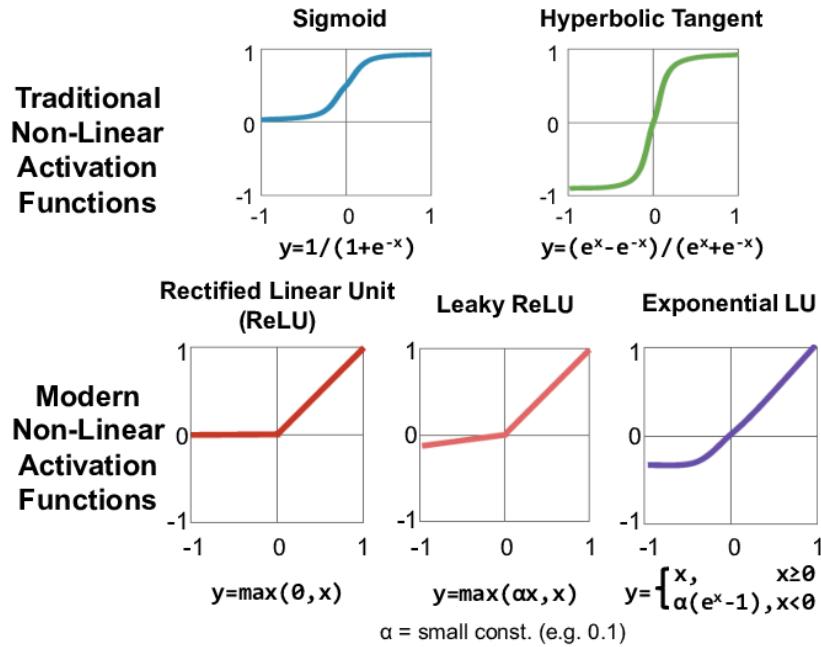


Abbildung 2.13: Illustration der wichtigsten Aktivierungsfunktionen.[3]

Beim Aufbau eines Recurrent Neural Network liegen einige Neuronen in einer Schleife, beispielsweise ein ganzer Layer. Diese Neuronen gehen die Input Matrix einen Merkmalsvektor nach dem anderen durch und berechnen dabei einen Output und einen Zustand, der bei der Bearbeitung des nächsten Merkmalsvektor eingerechnet wird.

Wie in der Abbildung 2.14 illustriert ist, kann die Schleife in einem Recurrent Neural Network entweder auf- oder ausgerollt sein. Wenn es aufgerollt ist, ist der Berechnungsgraph zyklisch, was die Berechnung der Ableitungen kompliziert. Beim Ausrollen wird ein Neuron, mit all seinen Parametern, kopiert für jeden Merkmalsvektor in der gegebenen Matrix. Bei der Berechnung der Ableitungen werden die Parameter in den einzelnen Neuronen verschiedene Ergebnisse erhalten, die Parameter sollen jedoch gleich bleiben, da sie ja eigentlich nur ein Neuron sind. Daher werden die einzelnen Steigungen vor dem Update der Parameter kombiniert, etwa indem der Durchschnitt genommen wird.

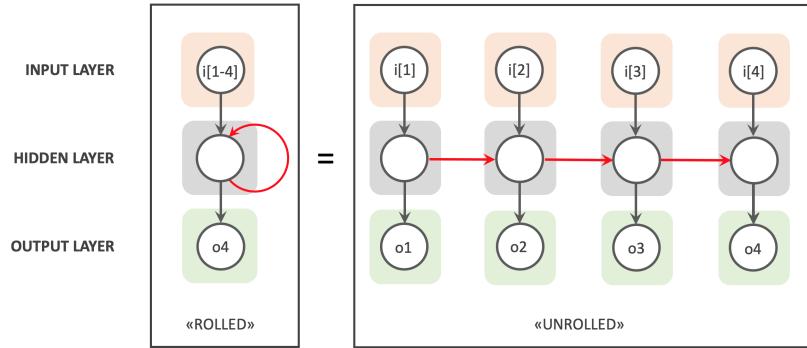


Abbildung 2.14: Illustration eines Recurrent Neural Networks.[1]

2.14 Python [CP]

Python ist eine High-Level, interpretierte, objektorientierte Programmiersprache, die in den letzten Jahren wegen einfacher, aber mächtiger Syntax beliebt wurde. Da Python sehr einfach zu erweitern ist und die Möglichkeit bietet Erweiterungen in C++ zu schreiben, um bessere Performance zu erzielen, kam es zu einer Explosion an frei downloadbaren Werkzeugen, die Packages genannt werden. Python ist für die Entwicklung von Künstlichen Intelligenzen mit Abstand am beliebtesten, daher gibt es auch extrem viele gute Packages, die die Entwicklung von Künstlicher Intelligenz einfacher machen.

2.14.1 Vorteile

Python ist sehr leicht zu lernen, bietet aber mächtige Werkzeuge, um alles Mögliche zu erreichen.

In C++ geschriebene Erweiterungen bieten dessen Performance in der simplen Syntax von Python.

Da Python interpretiert wird ist es platformunabhängig.

2.14.2 Nachteile

Python Code an sich ist eher langsam, dies kann jedoch durch das richtige Verwenden von guten Erweiterungen gemindert werden.

2.14.3 Alternative

2.14.3.1 C++

Während C++ noch mehr Performance bietet, ist jedoch der Code um einiges komplizierter und es gibt keine einfachen Erweiterungen wie bei Python.

2.15 Numpy [CP]

Numpy ist ein fundamentales und beliebtes Package für Python, dass sich mit effektiver und performanter Datenmanipulation beschäftigt. Es enthält unter anderem:

- Ein mächtiges N-dimensionales Array Objekt
- Komplexe Funktionen zum Rechnen mit diesen Arrays
- Verschiedenste Wege zufällige Nummern zu generieren

2.15.1 Vorteile

Wegen sehr guter Implementation ist es extrem performant und da Numpy auch institutionelle Partner hat, die zum Code beitragen, ist sichergestellt, dass die Implementationen auch korrekt sind und sich genau so verhalten, wie sie sollen.

2.16 Jupyter Lab [CP]

Jupyter Lab ist ein Web-basiertes, interaktives Interface, um Python Code zu entwickeln. Es baut dabei auf Jupyter Notebooks auf, welche für jedes Notebook einen Python Kernel starten, der Code interpretieren kann und temporäre Objekte wie Variablen speichert, und dem User die Möglichkeit bieten, Python Code in Kleinteile zu unterteilen und einzeln auszuführen, aber dabei trotzdem auf die gemeinsame Umgebung zuzugreifen.

Dies bringt den Vorteil, dass rasch Prototypen erstellt werden können, da, um etwas auszuprobieren, nicht das ganze Skript ausgeführt werden muss, sondern nur der betroffene Teil. Dadurch wird etwa Zeit beim Einlesen von Daten gespart, da diese nicht immer wieder neu geladen werden müssen.

Jupyter Lab bietet nun zusätzlich die Möglichkeit mehrere Notebooks im gleichen Browser Fenster offen zu haben und gibt außerdem Zugriff auf lokale Dateien anhand einer Verzeichnisstruktur im gleichen Fenster.

2.17 TensorFlow [CP]

TensorFlow ist ein von Google Brain entwickeltes open-source Framework zum effektiven Entwickeln, Trainieren und Bereitstellen von Künstlicher Intelligenz. TensorFlow selbst ist aus Performance Gründen in C++ implementiert, bietet aber APIs in verschiedenen Sprachen. Die Python API ist dabei die vollständigste, stabilste und beliebteste.

2.17.1 Graphen

Intern benutzt TensorFlow Berechnungsgraphen, die den Fluss von Daten vom Input eines Modells bis zum Output darstellen. Diese Graphen bestehen aus:

- Variablen, welche die trainierbaren Parameter des Modells darstellen
- Konstanten, welche konstante Werte in einer Berechnung darstellen
- Platzhaltern, welche die Inputs des Modells darstellen
- Operationen, welche Daten aus den obig genannten Datenquellen kombinieren und beliebig zusammengehängt werden können

Alle Daten im Graph werden dabei in Tensoren gespeichert. Ein Tensor ist eine Generalisierung von Vektoren und Matrizen. Tensoren werden durch ihren Rang beschrieben:

- Rang 0 ist ein Skalar
- Rang 1 ist ein Vektor
- Rang 2 ist eine Matrix
- usw.

Daher kommt auch der Name TensorFlow, die Tensoren fließen durch den Graphen und werden dabei immer weiter verändert.

Solche Berechnungsgraphen werden benutzt, da die verschiedenen Operationen auf der Best geeigneten Hardware ausgeführt werden können und da diese Graphen automatisch und schnell ableitbar sind, was zum Optimieren der Parameter gebraucht wird. Dieses automatische Ableiten ersetzt dabei Back Propagation.

Diese Graphen werden Objektorientiert im Code aufgebaut. Alle Variablen, Konstanten, Operationen und Platzhalter müssen selbst gesetzt und mit einander verbunden werden. Es werden zwar einige Hilfsmethoden angeboten, diese bleiben jedoch sehr Low-Level, daher muss sehr viel eigenständig implementiert werden.

2.17.2 Keras

Keras ist eine High-Level API, die für Python entwickelt wurde, um schnelle Implementation von Ideen zu ermöglichen. Seit der Version 2.0 von TensorFlow ist Keras ein Hauptbestandteil der Entwicklung mit TensorFlow, da es empfohlen wird, Low-Level TensorFlow Code in Keras Komponenten einzupacken und daher die Übersichtlichkeit des Codes zu erhöhen.

2.17.2.1 Layers

Keras Modelle sind aus so genannten Layers aufgebaut, die jeweils eine Schicht in einem Neuralen Netzwerk darstellen. Häufig genutzte Layers sind bereits implementiert, beispielsweise ein vollkommen verbundener Layer wird "Dense" genannt. Wie vorhin schon angedeutet können Layers auch eigens implementiert werden.

Die vordefinierten Layers bieten voreingestellte Parameter, die oft sehr gut passen, sie können jedoch auch leicht geändert werden, indem sie dem Konstruktor des Layers mitgeliefert werden.

Die verschiedenen Layers können entweder geradlinig mit einander verbunden sein, auch genannt "Sequential", oder wiederum selbst definiert über verschiedene Wege.

2.17.2.2 Vorteile

Die Keras API ist extrem leicht zu verstehen und zu benutzen, bietet aber trotzdem sehr viel Flexibilität in der Form von vielen Layers und Parametern, die eingestellt werden können.

Durch Kombination mit der Low-Level TensorFlow API ist noch mehr Flexibilität geboten, es können aber trotzdem High-Level Keras Features genossen werden.

2.17.2.3 Nachteile

Zumindest in der letzten 1.x Version von TensorFlow gab es noch einige Probleme mit der Integration von Keras, beispielsweise konnten exportierte Keras Modelle nicht richtig mit TensorFlow Serving, einem Teil von TensorFlow Extended TFX, benutzt werden. Diese Probleme scheinen jedoch ab Version 2.0 gelöst zu sein.

2.17.2.4 Alternative

Es könnte auch alles mit der Low-Level API geschrieben werden. Dies wäre jedoch extrem zeitaufwändig, es wäre Fehleranfälliger und der Code wäre weniger übersichtlich.

2.17.3 Vorteile

TensorFlow ist für das Trainieren von Modellen auf mehreren Computern optimiert und nutzt dabei vorhandene Hardware wie CPUs, GPUs und TPUs optimal aus. Daher ist es vor allem für sehr große Modelle mit Billionen von trainierbaren Parametern und Petabyte von Trainingsdaten sehr gut geeignet.

Da TensorFlow Zugriff auf sehr niedrigem Level bietet, kann es leicht für Forschungszwecke benutzt werden, und kann trotzdem von der Hardwareoptimierung profitieren. Da in der Forschung oft viele verschiedene Modelle ausprobiert werden

und nach Änderungen immer wieder neu trainiert werden ist auch hier Performance ein wichtiger Punkt.

Zudem bietet TensorFlow JavaScript Einbindung für etwa Web-Browser oder Node.js Server. Auf Mobil- und IoT-Geräten können mit TensorFlow Lite vortrainierte Modelle ausgeführt werden.

Mit TensorFlow Extended TFX werden Werkzeuge geboten für alles vom Einlesen von Daten bis zum Ausliefern und Laufen eines Modells.

Durch Tensorboard können Metriken und Graphen von verschiedenen Modellen einfach mit einander verglichen werden, ohne extra Code zum Visualisieren der Daten schreiben zu müssen.

2.17.4 Nachteile

Genau weil TensorFlow an sich so Low-Level ist, kann es für Anfänger im Feld Künstliche Intelligenz schwierig sein sich einzufinden. Dieser Nachteil wird jedoch komplett durch Keras aufgehoben.

2.17.5 Alternative

2.17.5.1 PyTorch

PyTorch ist ein weiteres open-source Framework für Python, hat jedoch weniger Werkzeuge als TensorFlow, die bei der Entwicklung und Auslieferung von Modellen. Der wichtigste Unterschied zu TensorFlow ist die Art, in der Berechnungsgraphen aufgebaut sind. Während diese Graphen in TensorFlow statisch definiert sind, werden sie in PyTorch dynamisch aufgebaut. Dies ist vor Allem bei Recurrent Neural Networks mit variabler Inputlänge hilfreich.

2.18 JSON [CP]

JSON ist die Abkürzung für Javascript Objekt Notation und ist ein äußerst beliebtes Format, um strukturierte Daten zu übersenden. Eine JSON Datei besteht



Abbildung 2.15: Beispiel für eine JSON Datei.[13]

grundsätzlich aus Name-Wert-Paaren. Der Name wird dabei in Hochkomma angegeben und der Wert kann einen der folgenden Typen haben:

- String, eine einfache Charakter-folge in Hochkomma gegeben
- Nummer, wird für Integer und Floats benutzt
- Boolean, ein simpler Wahr oder Falsch Wert
- null, um keinen Wert zu geben
- Array, eine Liste von JSON Werten
- Objekt, eine Liste von JSON Name-Wert-Paaren

In den meisten modernen Programmiersprachen gibt es eingebaute Unterstützung für das Einlesen, Erstellen und Verändern von JSON formatierten Daten.

Kapitel 3

Praktischer Teil

3.0.1 Aufbau [SD]

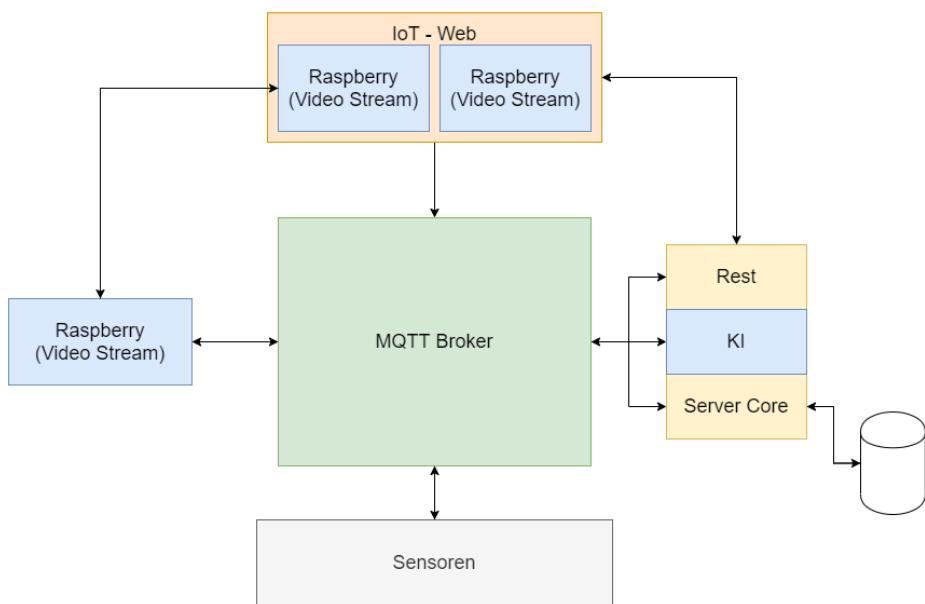


Abbildung 3.1: Architekut der Gesamtsystem. Der Raspberry Pi, Das 3d - Model welches ein Teil des IoT - Web Teil ist und die Künstliche Intelligenz, alle in Blau abgebildet, wurden in dieser Diplorarbeit entwickelt.

3.0.2 Bestandteile[SD]

- Künstliche Intelligenz KI (blau)
Vorhersage von Daten mittels Künstlicher Intelligenz.
- Raspberry (blau)
Dieser Streamt einen Live Feed einer Webcam.
- IoT - Web 3d Modell (blau)
Dieser Teil zeigt ein Interaktives 3d - Modell der HTL Leonding und zeigt aktuelle Messwerte der Sensoren an.
- IoT - Web Dashboard (orange)
Dieser Teil zeigt aktuelle Live Daten an, sowie weiter Informationen wie etwa die News aus dem CMS der HTL Leonding.
- Sensoren (Grau)
Co2, Licht, Temperatur, Lautstärke - Messungen. Diese werden im MQTT - Broker gespeichert.
- Server Core (Gelb)
Speicherung der Daten, Schnittstelle der einzelnen Teile des Projektes.
- Rest (Gelb): Restschnittstelle für das Versendet und Empfangen von Daten.

3.1 Raspberry Camera [SD]

3.1.1 Aufgabenstellungen

Für den Raspberry Pi welcher den Livestream sendet soll folgende Aufgabenstellung abdecken:

- Webcam - Stream
Es soll der aktuelle Livefeed einer Videokamera gestreamt werden.
- Webcam ein - / ausschalten
Videostream soll gestartet und gestoppt werden können. Wenn kein Videostream läuft soll die Webcam ausgeschaltet sein.
- Selbständige Integration
Beim Starten soll sich der Raspberry selbstständig in das IoT-System integrieren ohne einen Input eines Users.
- Mutli - Client - Streaming
Es darf beim Streaming keine Beschränkung für die Anzahl der aktiven Empfänger geben.

3.1.2 Aufgabenstellung: Webcam - Stream

Die beste Möglichkeit für die Umsetzung des Videstream schien WebRTC. WebRTC überzeugte dabei mit vielen Online Beispielen, welche ohne Installation direkt im Webbrower ausprobieren werden können. Die Qualität war dabei sehr zufriedenstellend und die Vorteile von WebRTC sprachen auch für die Verwendung von WebRTC.

3.1.2.1 WebRTC

Im ersten Anlauf für den Raspberry Streamer wurde das Packet aiortc verwendet. Aiortc ist eine Implementierung für WebRTC in Python.

Entwickelt wurde die Software nicht auf dem Raspberry, sondern auf einem normalen Computer bzw. Laptop. Der entwickelte Prototyp wurde zu Hause getestet,

dabei auch noch nicht auf dem Raspberry Pi. WebRTC lief dabei flüssig, stabil und auch ohne große Verzögerung und überzeugte uns für die Portierung auf den Raspberry Pi.

Im nächsten Schritt wurde die gesamte Applikation auf den Raspberry Pi portiert. Dabei wurden keine Änderungen am Source-Code vollzogen, jedoch musste aufgrund der ARM-Architektur auf die ARM-Kompatible Python verwendet werden. Diese unterscheidet sich nicht in Features. Bei der Installation des Package Aiortc gab es erste Probleme da dieses nicht für ARM vor kompiliert verfügbar war. Es musste auf dem Raspberry kompiliert werden.

Das erste Problem für diese Implementation, war das Schulnetz der HTL - Leonding. Dieses Netzwerk ist umfangreich und unterschied sich von dem ein Router - Netzwerk, in jenem die Applikation getestet wurde. Anfangs konnte kein Videostream innerhalb des Schulnetzwerks hergestellt werden. Dieses Problem konnte durch die Verwendung von TURN (siehe Kapitel WebRTC) gelöst werden, dabei litt die Qualität, subjektiv gesehen, nicht.

Das zweite Problem trat beim Testlauf des Raspberry mit WebRTC auf. Der Raspberry schien zwar nicht an seine Grenzen hinsichtlich Leistung zu stoßen, dennoch war der Videostream kaum bis gar nicht flüssig. Nach erfolgloser Suche wo sich dieses Bottleneck befand oder beheben können, musste der WebRTC - Anlauf verworfen werden.

3.1.3 Motion

Aufgrund der vielen bereits vorhandenen Lösung mit Motion im Internet, wurde versucht die Anwendung im zweiten Anlauf mittels Motion umzusetzen. Motion zeigt im ersten Test mit WebRTC eine schlechtere Performance. Um nicht dieselben Fehler wie beim WebRTC - Prototyp zu machen, wurde bei diesem Versuch gleich auf der Raspberry Pi Plattform entwickelt und getestet.

Anfänglich wurde mit Motion kein ruckelfreier Stream erzielt. Erst nach Einstellen vieler Parameter (siehe Kapitel Motion Konfiguration) wurde eine Zufriedenstellende Stream Qualität erreicht.

3.1.4 Aufgabenstellung: Webcam ein - und ausschalten

Die zweite Aufgabenstellung, welche die entwickelte Anwendung erfüllen sollte, war das, die Webcam nicht eingeschaltet ist, sofern gerade kein Client den Video - Stream sehen will. Da Motion eine Software ist, welche entweder gestartet ist und die Webcam streamt oder nicht, musste eine Lösung gefunden werden welche Motion auch starten und stoppen kann. Für diese Anforderung wurde ein Python Skript entwickelt.

Das Skript besteht aus folgenden Teilen:

- **HTTPs Server**

Für den HTTPs Server wurde das Packet aiohttp verwendet, welches Python um die Funktion eines HTTP Server erweitert. Zusammen mit einer SSL Verschlüsselung (siehe theoretischer Teil, Kapitel SSL/TLS), welches in der Standardbibliothek bereits zur Verfügung gestellt wird, wurde dieser Server abgesichert.

Der Server bietet einmal die Route `/start`, welche einen Start - Befehl an den Motioncontroller weitergibt, sofern dieser noch nicht bereits gestartet ist. Und einmal eine `/stop` Route. Das Skript weiß wie viele Empfänger gerade vorhanden sind. Sinkt die Anzahl der Empfänger auf 0, so wird ein Stop - Befehl an den Motion Controller gesendet.

- **Motion Controller**

Der Motion Controller ist zuständig für das Starten und Stoppen des Motion Daemon zuständig. Python besitzt die Möglichkeit Konsolenbefehle direkt auszuführen und da Motion eine Software ist welche über solche Befehle kontrolliert werden kann würde diese Möglichkeit verwendet.

- **MQTT Integration**

Für die Aufgabenstellung: Selbständige Integration findet das Skript beim Starten die IP - Adresse der Raspberry heraus und sendet diese dann an den konfigurierten MQTT Broker auf den konfigurierten Raum.

3.1.5 Sicherheit

Das IoT - Web Angular Projekt ist auf HTTPs gehostet, das bedeutet aber das nur HTTPs requests von dieser Seite weggeschickt werden dürfen. Somit muss auch der Raspberry Pi den Video - Stream und den HTTP Server verschlüsseln.

Wie das für den Python HTTP Server aussieht, wurde in dem Unterkapitel HTTPs Server der Aufgabenstellung Webcam ein - / ausschalten bereits beschrieben.

Für Motion war das ganze dann etwas umständlich. Nach Recherche haben wir herausgefunden das Motion ab Version 4.2. auch selbst eine Verschlüsselung anbietet. Diese Version war allerdings nicht mittels Paketmanager *apt* installierbar, sondern musste selbst installiert werden. Leider stießen wir wie viele andere Benutzer dieser Version auf das Problem das Streamen von Bildern nicht mehr funktioniert hat. Wir sind auf die Version 4.1.1. zurückgegangen und haben für eine Verschlüsselung den Reverse Proxy NGINX verwendet.

3.1.6 Konfiguration

3.1.6.1 Python-Controller

3.1.6.1.1 Raum

Der Raum, in dem sich der Raspberry Pi befindet, wird im Code konfiguriert. Eine Konfigurationsdatei wäre möglich gewesen doch aufgrund der wenigen Einstellungsmöglichkeiten wurde darauf verzichtet.

```
room = 'e581'
```

3.1.6.1.2 Mqtt Broker

Da das gesamte Projekt doppelt gehostet wird, um im Fall eines Ausfalls, trotzdem noch Verfügbar zu sein, kann auch die Url auf welcher der MQTT Broker (siehe Kapitel MQTT) läuft konfiguriert werden.

```
mqtt_url = 'vm103.h1-leonding.ac.at'
```

3.1.6.1.3 SSL Verschlüsselung

Für die SSL Verschlüsselung benötigt das Script ein Zertifikats - File und eine Schlüssel - File.

```
cert_path = 'cert.pem'  
key_path = 'key_cert.pem'
```

In der Konfiguration kann entweder ein relativer und absoluter Pfad angeben werden. Beim relativen Pfad ist zu beachten von wem das Script ausgeführt wird da unterschiedliche Benutzer unterschiedliche Startpfade besitzen können.

3.1.6.1.4 Motion

Die Anzahl der horizontal aufgenommen Pixel. Der Maximalwert ist dabei von der Kamera abhängig.

width 640 -> 1080

Die Anzahl der vertikal aufgenommen Pixel. Der Maximalwert ist dabei von der Kamera abhängig.

height 480 -> 800

Maximale Anzahl an Pixel welche aufgenommen werden kann.

framerate 30 -> 1000

Schwellwert für wie viele Pixel sich verändert müssen um eine Bewegung erkannt zu haben.

threshold 100 -> 0

Automatische Rauschunterdrückung.

noise_tune on -> off

Anzahl wie viele Bilder vor einer Bewegung aufgenommen und gesendet werden sollen. Diese werden erstmal gebuffered.

pre_capture 5 -> 0

Wie viele Bilder nach einer Bewegung noch aufgenommen werden sollen.

post_capture 900 -> 0

Anzahl an Sekunden die nach einer Bewegung noch aufgenommen werden. 0 steht für eine Lückenlose Aufnahme.

event_gap 60 -> 0

Die Qualität welche JPEG nach der Komprimierung noch haben soll.

quality 75 -> 90

Der Port auf dem der HTTP - Server läuft auf welchem das MJPEG Bild gehostet

wird.

stream_port 8081 -> 8881

Maximale Anzahl an gesendeten Bildern pro Sekunde im Stream.

stream_maxrate 1 -> 60

3.1.6.2 NGINX

```
server {  
    listen 443 ssl http2;  
    include snippets/self-signed.conf;  
    include snippets/ssl-params.conf;  
    location /mjpegs {  
        proxy_pass http://127.0.0.1:8881/;  
        proxy_read_timeout 1800;  
        proxy_connect_timeout 1800;  
        access_log off;  
    }  
}
```

- *listen 443 ssl http2.*

Gibt an das der NGINX - Worker auf den Standardport für HTTPS 443 hört, dabei verwendet er eine SSL Verschlüsselung und es wird das Protokoll HTTP2 verwendet, welches mit HTTP1 rückwärts kompatibel ist.

- *include snippets/self-signed.conf;*

Für die SSL Verschlüsselung benötigt man ein Zertifikat welches im Unterordner Snippets mit dem Namen self.signed.conf abgespeichert ist.

- *include snippets/ssl-params.conf;*

Für eine Verschlüsselung müssen bestimmte Parameter wie der Verschlüsselungsalgorithmus eingestellt werden. Diese sind im File ssl-params.conf abgespeichert und werden eingebunden.

```
• location /mjpegs {
    proxy_pass          http://127.0.0.1:8881/;
    proxy_read_timeout  1800;
    proxy_connect_timeout 1800;
    access_log off;
}
```

Mit Location wird ein Teilstück der URL beschrieben. Wenn ein Request auf `https://url-des-raspberry/mjpegs` nun erfolgt, leitet NGINX diesen an den internen Port 8881 weiter.

3.1.7 Bedienung

Um den Raspberry zu einem Video Stream Server zu konvertieren muss das Python Skript ausgeführt werden, vorausgesetzt alle benötigten Softwarepakete sind bereits installiert. Der Python Befehl für das einmalige Starten der Anwendung lautet:

```
python skript.py
```

3.1.7.1 Python Version

Es sollte darauf geachtet werden, dass eine Python Version ab 3.0 verwendet wird. Die Version von Python kann mittels des Kommandos `python -v` herausgefunden werden. Bei älteren Versionen kann dieser Befehl zu einem Fehler führen da dieser erst mit Version 2.5 eingeführt wurde. Wenn die Ausgabe der Versionsnummer mit 2 beginnt, muss entweder der globale Python Interpreter geändert werden oder man ersetzt im Befehl `python skript.py "python"` mit `python3` oder `py -3`.

Python läuft nun und gibt etwaige Informationen auf der Konsole aus. Um das Programm zu stoppen, kann die Tastenkombination STR + C benutzt werden. Alternative könnte der Python Prozess über den Taskmanager gestoppt werden oder die Konsole geschlossen werden.

3.1.7.2 Automatischer Start beim Hochfahren

Um nicht nach jedem Neustart des Systems das Skript neu zu starten wurde ein Systemd Service verwendet. Systemd kann man mit dem Autostart von Windows vergleichen. Doch es bietet weitere Optionen wie das Neustarten des Services, oder eine Angabe nach welchen Komponenten dieser Service gestartet werden soll. Letzter ist sehr wichtig da die Anwendung den Netzwerk - Service benötigt um seine IP - Adresse herauszufinden.

3.1.7.2.1 Systemd

Um einen Systemd - Service zu erstellen, muss eine Datei mit der Endung .service im ordner /lib/systemd/system/ angelegt werden. Hierzu werden Superuser Rechte benötigt. Der Service ist inaktiv.

Hinweis für alle mit systemctl beginnenden Befehle werden Superuser - Rechte benötigt.

Um einen Systemd - Service zu starten.

```
systemctl start SERVICENAME
```

Der Service ist zwar jetzt gestartet, wird aber nicht automatisch beim Hochfahren gestartet. Dieser Modus ist für das Ausprobieren eines Services gedacht. Gestoppt werden kann dieser mit:

```
systemctl stop SERVICENAME
```

Um seinen Service beim Start anzuhängen, muss dieser aktiviert werden.

```
systemctl enable SERVICENAME
```

Will man den Service nicht mehr beim Starten ausführen so kann dieser Service vom Autostart entfernt werden.

```
systemctl disable SERVICENAME
```

Jeder Service ist ein Daemon Prozess. Der User sieht den Status nicht direkt, sondern muss mit Hilfe von Systemd abgerufen werden.

```
systemctl status SERVICENAME
```

3.2 IoT - Web [SD]

Das IoT - Web Projekt ist jener Teil der für den Benutzer zugänglich ist. Dieser wurde mittels Angular umgesetzt. Dabei wurden unter anderem auch Technologien wie Three.js und MQTT verwendet.

3.2.1 Aufgaben

- Live - Daten

Im Angular Frontend sollen die aktuell gemessenen Werte der Sensoren angezeigt werden.

- Video Stream

Im Interface soll die Möglichkeit bestehen eine Live - Stream anzusehen.

- Filter

Die Räume der Stockwerke sollen für einen ausgewählten Sensor entsprechend ihres aktuellen Messwertes eingefärbt werden.

3.2.2 3d Model

Für die Erstellung eines interaktiven 3d - Model der Schule wurde Three.js verwenden. Für nähere Informationen siehe Kapitel Three.js. Mittels der OrbitalControll von Three.js kann das Model rotiert, verschoben, verkleinert und vergrößert werden.

In diesem Model ist es möglich einen Querschnitt der Stockwerke zu erhalten. Die Stockwerke sind entweder über das Menü auswählbar oder durch klicken auf das Model. Falls ein Stockwerk durch Klicken im 3d - Model ausgewählt wird, muss das Model sicherstellen, das in diesem Fall auch wirklich ein Stockwerk ausgewählt wurde und nicht ein Raum. Den auch Räume können angeklickt werden, mithilfe eines Raycaster wird ermittel auf welchen Block als erstes getroffen wird. Ist dieser Block, ein Raum so wird der Raum aktiviert und man erhält Live - Daten dieses Raumes, handelt es sich dabei aber um ein Stockwerk, so wird der Querschnitt dieses Stockwerks angezeigt. Bei einem Querschnitt werden alle darüber liegenden Stockwerke ausgeblendet.

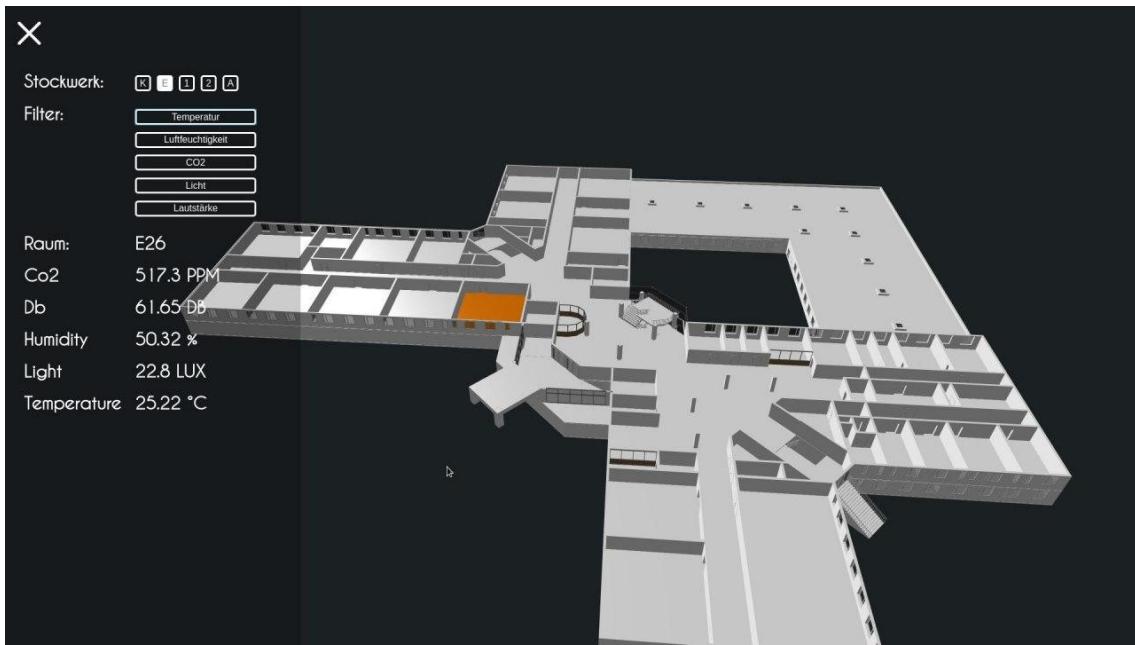


Abbildung 3.2: 3d - Model mit aktiven Raum

Wird ein Raum angeklickt, erhält man im Menü die aktuellen Live - Werte des ausgewählten Raumes. Diese Daten werden von den Sensoren gesammelt und mittels MQTT in das IoT - Projekt gespeist.

Mithilfe von Observer und dem async Pipe - Operator wurde dies erzielt.

```
 {{(measurementTypeAndValue?.measurement | async)?.value }}  
 {{ (measurementTypeAndValue?.measurement | async)?.unit}}
```

Abbildung 3.3: MQTT - HTML Code

```

return new Observable<Measurement>(observer => {
  this.observe(
    `htlleonding/${area.name}/${section.name}/${position}/${sensor}/#`
  ).subscribe((message: IMqttMessage) => {
    const json = JSON.parse(message.payload.toString());
    const m: Measurement = new Measurement();
    m.value = json.value;
    m.timestamp = json.timestamp;
    m.unit = munit;
    observer.next(m);
  });
});

```

Abbildung 3.4: MQTT - Typescript Code

Der Code sendet dabei einen Request an den MQTT Broker, dabei bekommt er sofort den letzten gespeichert werden. Sobald der Broker eine neue Messung erhält, sendet er diese weiter an den Web Client und wird an den Observer weiter gegeben.

Um Livewerte auch in der View anzeigen zu können, kann der Pipe - Operator “async” eingesetzt werden, dieser wertet ein Statement nicht einmal aus, sondern öfters, um die Daten am aktuellen Stand zu halten. Um sicherzustellen, dass es eine Messung gibt, wird der “?” Operator eingesetzt. Dieser prüft, ob eine Messung vorhanden ist, und nur wenn eine Messung vorhanden wird das restliche Statement, sprich die Abfrage des Messwertes und die Einheit, weiter ausgeführt.

Ist für diesen Raum eine Webcam registriert so wird im Menü zusätzlich ein Start und Stop - Knopf für den Video - Stream angezeigt. Erst wenn Start gedrückt wird, sieht man einen Video - Stream.

Für eine schnelle Übersicht über die Messwerte in mehreren Räumen wurden Filter entwickelt. Um die Benutzerfreundlichkeit zu steigern wurden die Farben entsprechen ihres Typen gewählt:

- Temperatur, von Blau kalt bis Rot warm
- Luftfeuchtigkeit, von 0% Weiß bis 100% Blau
- Co₂, 500 Grün bis 800 Rot
- Licht, Weiß aus, Gelb an
- Lautstärke, 50 DB Grün bis 100 DB Rot

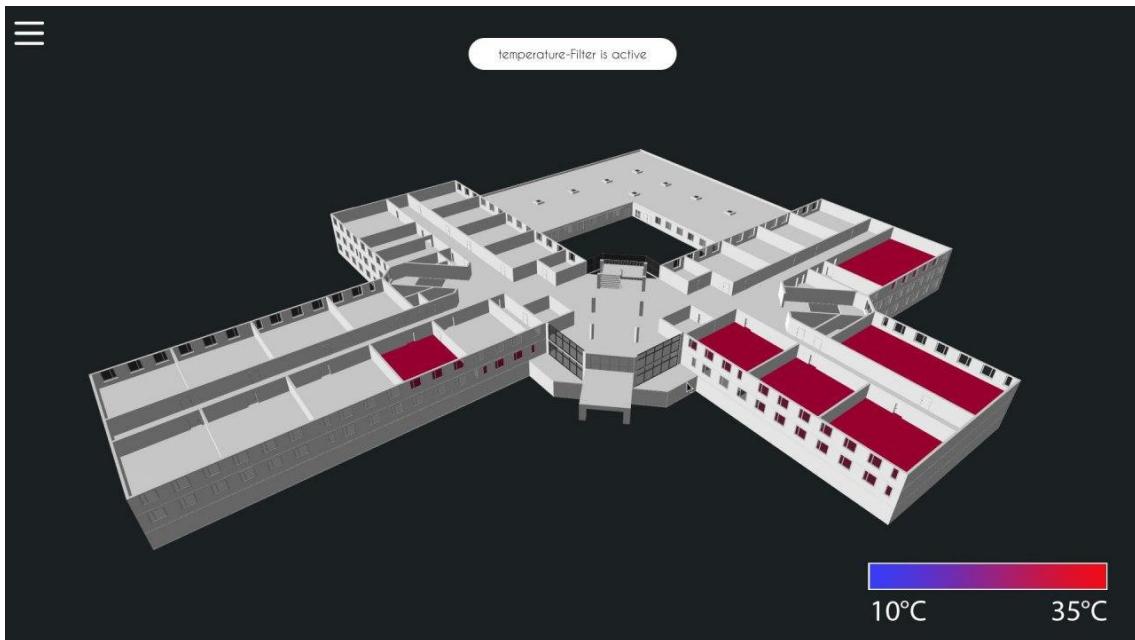


Abbildung 3.5: 3d - Model mit aktivierter Filter

Für eine Steuerung von Außen besteht das MQTT - Topic '*htlleodning/3dmodel/control/room*' auf dieses Topic kann ein Raum geschickt werden welcher dann im 3d - Modell geöffnet wird. Natürlich wird auch das entsprechende Stockwerk ausgewählt und aktiviert.

Um zu zeigen, dass es sich bei dem Model nicht nur um ein Bild handelt, sondern um ein interaktives Schulgebäude, fängt das Model nach einer bestimmten Zeit an sich zudrehen.

3.2.3 IoT - Web Core

Der Core Teil übernimmt die Logik für den Datenverkehr für das 3d - Model und dem Dashboard. Der Datenverkehr beschränkt sich dabei nicht nur auf MQTT Daten, sondern auch auf andere APIs wie die Linz AG API zur Abfrage des Straßenbahnpans. Der Zugriff auf das Content - Management - System.

3.2.4 Dashboard

Im Dashboard ist ein Interaktives 2d Model der Schule, wenn man auf einen Raum klickt werden die aktuellen Live - Daten angezeigt.

Im Dashboard werden aktuelle Messwerte, der Sensoren und der Wetterstation am Dach der HTL Leonding, angezeigt. Des Weiteren sind aktuelle News aus dem Content - Management - System der HTL Leonding zu sehen.

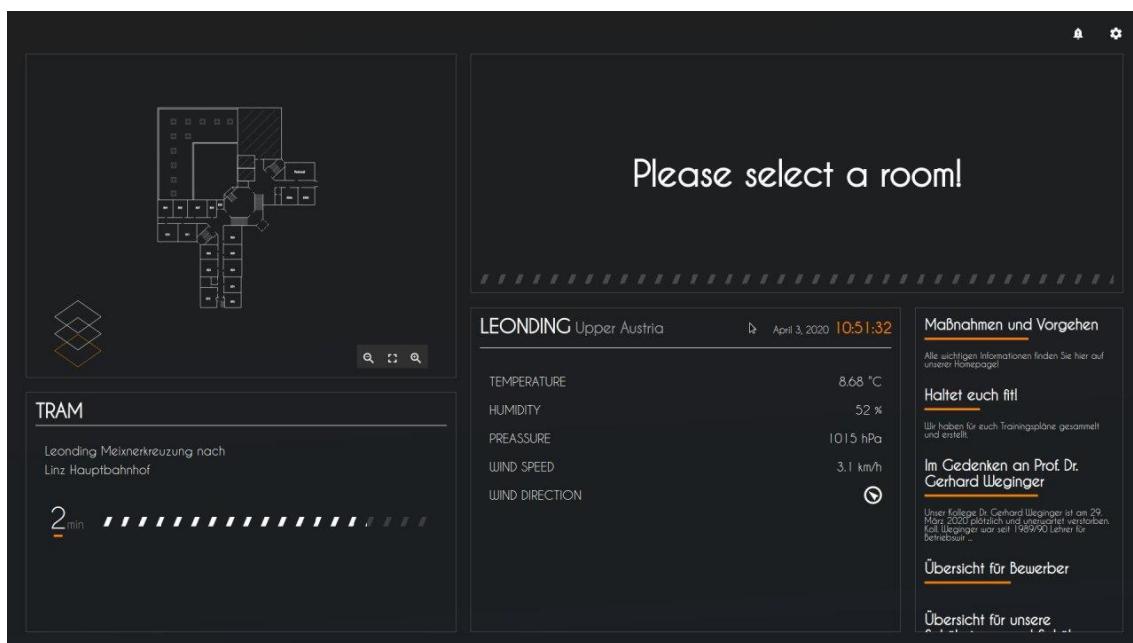


Abbildung 3.6: 3d - Model mit aktivierter Filter

3.3 Installation [SD]

3.3.1 Raspberry Pi Video Stream

Es werden folgende Gegenstände benötigt:

- Raspberry Pi Version 3 oder neuer
- SD - Karte mit 16 Gigabyte oder mehr
- Webcam
Hierfür kann entweder eine USB Webcam oder ein Raspberry Kameramodul verwendet werden.

Für die Installation braucht man zusätzlich noch:

- Einen Computer / Laptop, mit SD - Karten Slot
- Eine Tastatur
- Ein Bildschirm

3.3.1.1 Raspberry Installation

Der Raspberry hat Standardmäßig keine Software installiert. Bis auf wenige Ausnahmen besitzt der Raspberry beim Erwerb auch kein Speichermedium. Deswegen muss das Betriebssystem selbst installiert werden.

3.3.1.2 Installation vom Betriebssystem

Für diesen Schritt muss eine mit dem Raspberry kompatible Linux Version heruntergeladen werden. Die beste Wahl hierfür ist Raspbian eine Linux Version, welche extra für den Raspberry entwickelt wurde, es bestehen aber auch noch andere Optionen wie etwa Ubuntu. Um dieses Betriebssystem auf dem Raspberry verwendet zu können muss es auf einer SD - Karte installiert werden. Hierfür empfehlen sich Tools wie Rufus oder Etcher.

3.3.1.3 Raspberry Pi Aufsetzen

Für das erste Mal starten benötigt der Raspberry einen Bildschirm und eine Tastatur. Sobald das Aufsetzen am Raspberry abgeschlossen ist, können die restliche Installationsschritte entweder in der GUI oder mithilfe von SSH abgeschlossen werden. SSH Verbindung entlasten die Ressourcen des Raspberry Pi, insbesondere den RAM.

3.3.1.4 Python

Auf vielen Linux Versionen ist Python bereits vorinstalliert. Dabei sollte man aber sicherstellen das eine Version von 3.0 oder höher installiert ist.

Für das Pythonskript wird das Paket aiortc benötigt welches über den Paketmanager von Python *pip* installiert werden kann. Bevor aiortc installieren werden kann, müssen noch die Abhängigkeiten dieses Paketes installiert werden.

```
apt install libavdevice-dev libavfilter-dev libopus-dev  
      \ libvpx-dev pkg-config.  
pip install aiortc
```

3.3.1.5 Venv

Unter Python besteht die Möglichkeit Pakete nicht global, sondern nur in einer abgekapselten Umgebung genannt Venv, zu installieren. Diese abgekapselte Umgebung bietet den Vorteil, Versionskonflikte zu vermeiden. Venv kann installiert und aktiviert werden, ist aber nicht zwingend notwendig.

3.3.1.6 SSL Zertifikat

Für die Verschlüsselung des HTTPs - Server im Skript muss noch ein SSL - Zertifikat erstellt werden. Es wird empfohlen die Befehle im gleichen Ordner wie das Python File auszuführen, um Probleme mit Pfaden zu verhindern.

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem  
           -out cert.pem -days 365
```

Der oben angeführte Befehl erstellt ein SSL - Zertifikat und dem dazugehörigen Schlüssel. Wenn nur diese 2 Files verwendet werden, muss beim Starten des Skriptes die Passphrase eingegeben werden, um dies zu verhindern, kann ein Schlüsselfile mit gespeicherter Phrase erstellt werden:

```
openssl pkey -in key.pem -out key_cert.pem
```

3.3.1.7 Motion

Motion kann entweder über den normalen Paketmanager oder selbst installiert werden. Der Paketmanager hat nicht immer die aktuellste Version von Motion, aus diesem Grund kann es sich unter Umständen empfehlen die neuste Version manuell zu installieren.

Motion via apt-get

```
sudo apt-get install -y motion
```

-y beantwortet alle etwaigen Fragen mit Ja.

3.3.1.8 Nginx

Nginx muss für eine Verschlüsselung für Motion installiert werden.

```
apt-get install nginx
```

Motion ist dabei auch ein Systemd Service.

```
systemctl status nginx
```

Bei Nginx wird die default.conf Datei verändert. Alternative könnte auch eine neue Konfigurationsdatei angelegt werden.

Option 1: default.conf bearbeiten:

```
sudo nano /etc/nginx/conf.d/default.conf
```

Option 2: Neue Konfigurationsdatei

Konfigurationsdatei anlegen:

```
sudo touch /etc/nginx/sites-enabled/motion.local
```

Konfigurationsdatei bearbeiten

```
sudo nano /etc/nginx/sites-enabled/motion.local
```

Diese Datei muss auf die Konfiguration siehe Konfiguration abgeändert werden.

3.3.1.9 NGINX Sicherheit:

Erstellen eines SSL - Zertifikat:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048  
-keyout /etc/ssl/private/nginx-selfsigned.key  
-out /etc/ssl/certs/nginx-selfsigned.crt
```

Für NGINX muss noch konfiguriert werden welche SSL Verschlüsselung verwendet wird:

```
nano /etc/nginx/snippets/ssl-params.conf
```

Änderungen von NGINX können Validiert werden mit:

```
nginx -t
```

Wenn dies erfolgreich war kann nun NGINX neu gestartet werden:

```
service nginx reload
```

3.3.2 Installation auf der Virtuellen Maschine

3.3.2.1 Docker - Compose

Um die benötigte Software auf dem Server minimal zu halten wurden alle Teile der Diplomarbeit gedockert und zu einem großen *docker-compose.yml* zusammengefügt. Dieses File greift auf eine schulinterne Registry zu um sich Docker - Images herunterzuladen. Auf der Virtuellen Maschine muss deswegen nur *Docker* und *docker-compose* installiert werden. Um das Projekt zu starten:

```
docker-compose up --build -d
```

3.4 System Architektur - Künstliche Intelligenz [CP]

Das System zur Vorhersagung, ob in einem Raum ein Fenster offen ist oder nicht, ist eine Erweiterung des derzeit bestehenden System. Es nutzt dabei den bestehenden MQTT Broker zur Kommunikation zwischen den Bestandteilen des Systems und zur Bereitstellung der Vorhersagen zum Rest des bestehenden Systems.

Das trainierte Modell, die Künstliche Intelligenz, läuft in einem Docker Container auf der selben Maschine als der Broker, welche eine GPU beinhaltet, wodurch das Vorhersagen von Fensterzuständen erheblich beschleunigt wird.

Da beim bestehenden System bei jeder Änderung eines Messwertes in einem Raum ein Update gesendet wird, macht es keinen Sinn direkt auf diese Updates zu warten und für jedes Update eine einzelne Vorhersage zu treffen. Die Häufigkeit der Updates bei mehreren Räumen würde die Künstliche Intelligenz vermutlich überbeanspruchen. Außerdem kann die Hardware besser genutzt werden, wenn mehrere Vorhersagen gleichzeitig getroffen werden.

Aus diesem Grund wurde ein Topic am MQTT Broker festgelegt, welches zum "aufwecken" der Künstlichen Intelligenz dient. Wenn eine Nachricht an dieses Topic geschickt wird, reagiert die Künstliche Intelligenz indem sie für alle Räume den Status deren Fenster vorhersagt und an deren Topics am Broker bereitstellt.

Damit diese Nachricht zum Aufwecken der Künstlichen Intelligenz nicht immer manuell geschickt werden muss, wurde ein weiterer Docker Container erstellt, dessen einzige Aufgabe es ist, in einem festgelegtem Intervall eine Nachricht an dieses Topic zu senden und damit die Künstliche Intelligenz aufzuwecken.

Dieses Design des Systems bietet auch die Möglichkeit in einer Frontend Applikation den User die Künstliche Intelligenz manuell aufwecken zu lassen. Diese Option könnte vor Allem nützlich sein wenn Server Ressourcen gespart werden wollen, indem Vorhersagen nur getroffen werden wenn der User diese beantragt.

In der Entwicklung der Künstlichen Intelligenz mussten, wie auch zu erwarten, viele verschiedene Modelle trainiert und Ansätze geändert werden. Da die benötigte Hardware, die GPU, die für schnelles Training nötig ist nur am Server zur Verfügung steht, macht es das Entwickeln am privaten Computer sehr mühsam.

Daher, um die Entwicklung zu vereinfachen, wurde ein separater Docker Container genutzt, welcher Python und das TensorFlow Python Package mit GPU Support enthält. Zudem dient dieser Container als ein Jupyter Lab Server der von Außen

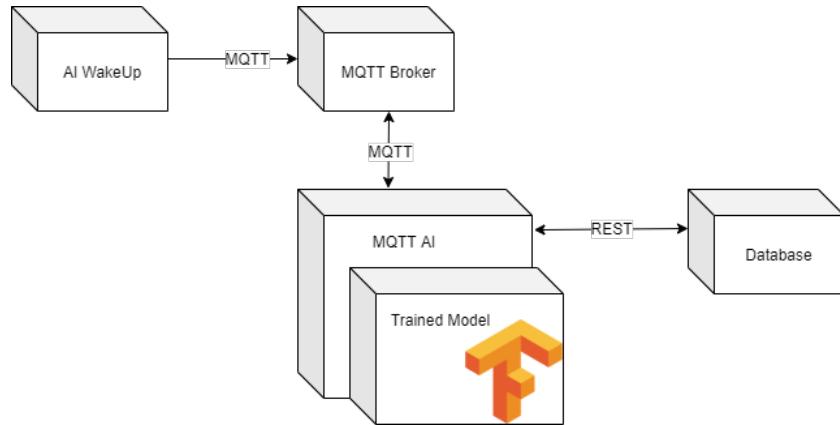


Abbildung 3.7: Illustration der System Architektur

erreichbar ist.

Diese Lösung bietet nun eine Entwicklungsumgebung mit allen nötigen Tools die bequem mit dem Internet Browser erreicht werden kann. Zudem macht Jupyter Lab das Visualisieren von Daten um einiges leichter, was bei der Entwicklung einer Künstlichen Intelligenz extrem wichtig ist. Diese Visualisierungen und auch alle anderen Ausgaben der Code Zellen in einem Notebook persistieren in dessen File, dies kann zusammen mit dem Markdown Support in Notebooks genutzt werden um Präsentationen vorzubereiten die etwa Code erklären und dessen Ausgaben schon vorhanden sind.

Der GPU Support in Docker ist derzeit noch sehr limitiert. Nach dem installieren von GPU Support in Docker, wird der extra Parameter ”–gpus all” beim Befehl ”docker run” bereitgestellt. Dieser Parameter macht alle GPUs des Systems im Docker Container zugänglich. Leider funktioniert dieser Parameter nur bei ”docker run”, welches einen neuen Container für ein Image erstellt. Daher können die Container welche Zugang zur GPU benötigen, der Container für die Künstliche Intelligenz und der Container zur Entwicklung, nicht im Docker Compose automatisch gestartet werden.

Als Lösung, bis offizieller GPU Support für Docker Compose veröffentlicht wird, wurden zwei Shell Scripts erstellt die manuell ausgeführt werden müssen und jeweils einen der zwei betroffenen Container mit allen benötigten Parametern startet.

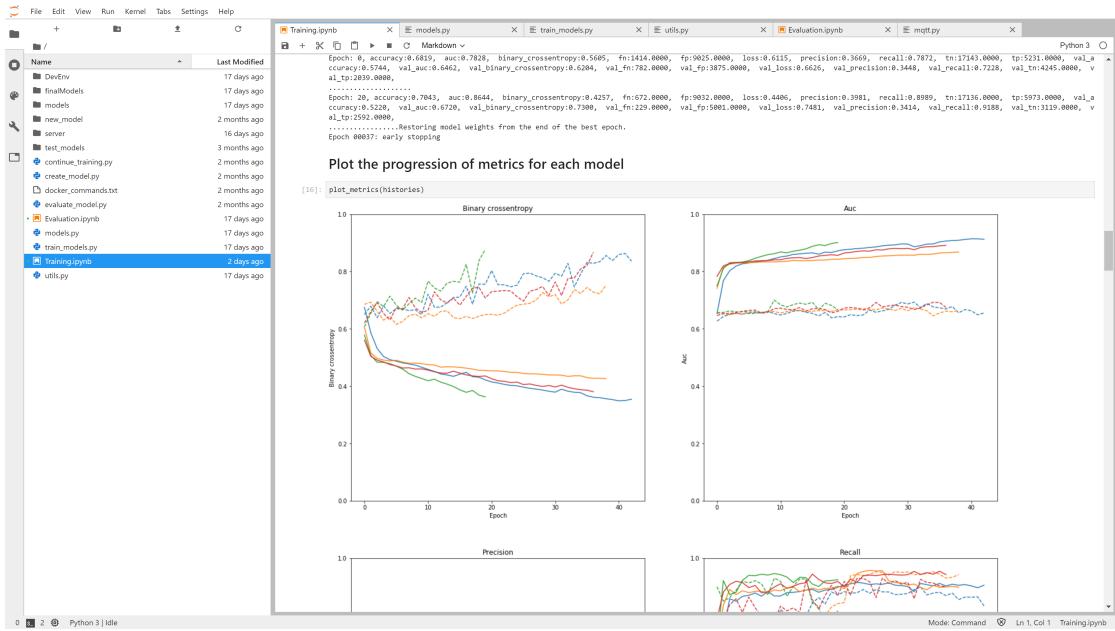


Abbildung 3.8: Ein Lupyter Lab Notebook

3.4.1 Datensammlung

Zum Trainieren der Künstlichen Intelligenz werden natürlich Messdaten benötigt, von denen das Modell lernen kann. Das Sammeln dieser Daten für einen Raum war die Aufgabe einer weiteren Diplomarbeit, durchgeführt von Luis Noisternig.

Es werden 5 Sensoren gemessen, 3 davon im Raum, welche in Zukunft in allen Klassenräumen vorhanden sein sollten, und 2 Sensoren außerhalb des Gebäudes, plus der Status der Fenster im Raum gemessen durch Schließkontakte an den Fenstern.

Die gemessenen Sensorwerte sind:

- CO₂ im Raum
- Luftfeuchtigkeit im Raum
- Luftfeuchtigkeit außerhalb des Gebäudes
- Temperatur im Raum
- Temperatur außerhalb des Gebäudes

```
{
  "object": [
    {
      "co2Indoor": 397.5,
      "humidityIndoor": 25,
      "humidityOutdoor": 37,
      "temperatureIndoor": 21.4,
      "temperatureOutdoor": 13.5,
      "timestamp": "Fri, 27 Mar 2020 17:16:26 GMT",
      "window1open": false,
      "window2open": false,
      "window3open": false,
      "windowOpen": false,
      "windowOpenCount": 0
    },
    {
      "co2Indoor": 397,
      "humidityIndoor": 25,
      "humidityOutdoor": 37,
      "temperatureIndoor": 21.4,
      "temperatureOutdoor": 13.4,
      "timestamp": "Fri, 27 Mar 2020 17:23:09 GMT",
      "window1open": false,
      "window2open": false,
      "window3open": false,
      "windowOpen": false,
      "windowOpenCount": 0
    }
  ],
  "success": true
}
```

Abbildung 3.9: Beispiel für <http://vm05.htl-leonding.ac.at/data/all?max=2>

Diese Messdaten werden in einer Datenbank gespeichert und sind durch eine REST Schnittstelle abrufbar unter der folgenden URL:

<http://vm05.htl-leonding.ac.at/data/all>

Diese URL liefert alle Messpunkte in der Datenbank sortiert nach deren Zeitstempel. Wenn nicht alle Messpunkte erwünscht sind kann auf die letzten n beschränkt werden mit dem Query Parameter "max":

<http://vm05.htl-leonding.ac.at/data/all?max=2>

Diese Schnittstellen geben ein JSON Objekt zurück, welches wiederum ein Array namens "object" enthält, in dem die Messpunkte als einzelne JSON Objekte gespeichert sind.

Vorsicht ist geboten da kein strenges Schema vorhanden ist, daher können Sensoren bei Messpunkten fehlen oder keinen Wert haben.

3.4.2 Installieren und Starten

Zuerst muss sichergestellt werden, dass alle nötigen Dateien lokal vorhanden sind. Diese könne in einem GitLab Repository an der folgenden URL gefunden werden:

<https://gitlab.com/sdanninger/iot>

Der erste Schritt ist es, das Gesamtsystem mit dessen docker-compose File zu starten.

Der MQTT Client, welcher die künstliche Intelligenz regelmäßig "aufweckt", wird bereits im docker-compose des Gesamtsystems gestartet.

Der MQTT Client, welcher die künstliche Intelligenz enthält, kann mit dem Shell-Skript "startAI.sh" gestartet werden. Dieses Skript erwartet ein gespeichertes Modell, in einem Format welches später noch beim Abspeichern eines Modells erklärt wird, an dem Pfad "/home/iot-team/iotai/model" zu sein. Ist dies nicht der Fall muss eines an den Pfad bewegt werden oder der Pfad im Skript geändert werden.

Zum starten der Entwicklungsumgebung, dem Jupyter Lab Server, muss das Shell-Skript "startDevEnv.sh" ausgeführt werden. Dieses Skript versucht den Ordner "/home/iot-team/iotai/source" online zugänglich zu machen. Ist der Source Code an einem anderen Pfad, muss er entweder zu diesem Pfad verschoben oder der Pfad im Skript geändert werden.

Sollte eines der Skripten nicht ausführbar sein muss zuerst folgender Befehl ausgeführt werden:

```
chmod +x #Skript#
```

Wobei #Skript# mit dem Namen des jeweiligen Skriptes ersetzt werden muss.

3.5 Künstliche Intelligenz [CP]

Dieser Abschnitt befasst sich mit allen Schritten nötig zum Erstellen einer Künstlichen Intelligenz und wie diese in der Praxis umgesetzt wurden.

Wie bereits erwähnt wurde die Künstliche Intelligenz in einer Jupyter Lab Umgebung entwickelt. Diese kann unter der folgenden URL erreicht werden, aus Sicherheitsgründen wird jedoch ein Passwort benötigt welches im Dockerfile für das Image des Containers festgelegt wird.

<https://vm103.htl-leonding.ac.at/aidev>

Dort können für spätere Referenz beziehungsweise zur Einleitung von Nachfolgern zwei Notebooks gefunden werden, welche das Trainieren und Evaluieren eines Modells durchlaufen mit dazugehörigen Kommentaren.

3.5.1 Vorgehensweise

Egal welche Art von Modell trainiert werden soll, der konzeptuelle Ablauf bleibt im etwa gleich.

3.5.1.0.1 Modell definieren Zuerst muss eine Architektur für das Modell erstellt werden, welche den gegebenen Anforderungen entspricht.

3.5.1.0.2 Daten einlesen Die gegebenen Messdaten müssen aus einer Quelle eingelesen werden.

3.5.1.0.3 Daten formatieren Die eingelesenen Daten entsprechen oft noch nicht dem benötigten Format für die gewählte Modell Architektur und müssen daher angepasst werden.

3.5.1.0.4 Daten aufteilen Um eine Evaluierung des Modells zu ermöglichen, müssen die Daten geteilt werden in Training- und Test-Datensätze.

3.5.1.0.5 Modell trainieren Das derzeitige Modell wird anhand der gegebenen Trainingsdaten trainiert und zeichnet dabei verschiedenste Metriken auf.

3.5.1.0.6 Bewerten und Verändern Das Modell wird anhand dem Verhalten der Metriken während dem Trainieren bewertet und Veränderungen werden anhand dieser Bewertung gemacht.

3.5.1.0.7 Evaluieren Wenn akzeptable Performance beim Training erzielt wurde, wird das Modell anhand des Test-Datensatzes evaluiert.

3.5.1.0.8 Modell speichern Zuletzt wird das erzeugte Modell in einem Format gespeichert, dass später im laufenden System leicht geladen werden kann.

3.5.2 Modell

Wie erwähnt wird zuerst das Modell definiert. Hierbei kommen viele Fragen auf; einige davon leichter wie: "Wie soll der Input und Output aussehen", andere hingegen schwieriger wie die Anzahl und Art der Layer im Netzwerk, wie viele Neuronen jeder dieser Layer beinhalten soll, welche Aktivierungsfunktionen die Layer haben sollen und so weiter.

Im Feld Machine Learning gibt es extrem viele solche Entscheidungen die oft arbiträr entschieden werden. Großteils kann sich nur auf Erfahrung verlassen um über einige Prototypen hinweg zum Ziel zu gelangen.

Um das erste Modell zu definieren muss zuerst die gegebene Problemstellung analysiert werden. Die folgenden Paragraphen zeigen diese Analyse und beschreiben welche Erkenntnisse daraus gezogen wurden anhand der Problemstellung dieser Diplomarbeit.

Eine Vorhersage anhand eines einzelnen Messpunktes wird vermutlich nur limitiert Erfolg haben, da das Modell nichts vom Verlauf der Messwerte weiß. Um dem Modell mehr Informationen zu liefern sollte der Input eher eine Sequenz von zeitlich sortierten Messpunkten sein. Eine Art von Neuronalen Netzwerken die sich mit zeitlichen Abfolgen beschäftigt sind Recurrent Neural Networks, als im theoretischen Teil besprochen. Diese Erkenntnis liefert die Antwort darauf, welche Art von Layers zu benutzen und welche Form der Input nehmen wird.

```

smallLSTM = tf.keras.Sequential([
    layers.LSTM(10, input_shape=(50, 5)),
    layers.Dense(1, activation='sigmoid')
])

smallGRU = tf.keras.Sequential([
    layers.GRU(10, input_shape=(50, 5)),
    layers.Dense(1, activation='sigmoid')
])

```

Abbildung 3.10: Die ersten definierten Modelle

Es ist nicht sinnvoll die gesamte Sequenz an Messpunkten als Input zu liefern, nur um den Fenster Status beim letzten Messpunkt vorherzusagen. Die wichtigen Veränderungen in den Messwerten sind kurz vor dem zu vorhersagenden Zeitpunkt, das bedeutet, es kann ein Limit n gesetzt werden und nur die neuesten n Messpunkte werden als Input gegeben. In diesem Fall wurde $n = 50$ gewählt.

Es soll vorhergesagt werden, ob in einem gegebenen Raum ein Fenster offen ist. Das heißt es gibt 2 Klassen als Output, "Offen" und "Geschlossen", und die Problemstellung kann als binäre Klassifikation benannt werden.

Mit diesen Erkenntnissen und Annahmen kann schon einiges festgelegt werden, wie die folgenden Paragraphen zeigen.

Der Input des Modells ist eine Matrix der Form $(50, 5)$ für jede Vorhersage, da 5 Sensoren gegeben sind und die letzten 50 Messpunkte mitgegeben werden. Modelle berechnen jedoch normalerweise mehrere Inputs gleichzeitig um die Hardware vollständig auszunutzen, so genanntes "batching". In TensorFlow wird das Modell automatisch mit einer extra Dimension beim Input und Output, die für batching genutzt wird. Daher ist der echte Input ein Tensor der Form $(None, 50, 5)$, wobei "None" für eine beliebige Größe steht.

Der letzte Layer beinhaltet nur einen Neuronen der wie bei Logistischer Regression besprochen die Wahrscheinlichkeit gibt, dass ein Fenster offen ist. Mit batching führt dies zu einem Ouput der Form $(None, 1)$.

Um ein Recurrent Neural Network zu erstellen muss mindestens ein Long-Short-Term-Memory LSTM oder Gated-Recurrent-Unit GRU Layer im Modell enthalten sein. Meist wird mit sehr kleinen Modellen angefangen, daher kann dieser Layer fürs Erste auf eine Größe von 5 bis 10 Neuronen gesetzt werden.

Damit ist das erste Modell auch schon definiert und kann trainiert werden. Jedoch ist es sehr mühsam immer ein einzelnes Modell zu definieren, bewerten und verbessern und das Vergleichen mit anderen Modellen ist aufwändig, daher werden gleich beispielsweise 5 Modelle definiert und zusammen bewertet und verbessert. Der ständige Vergleich zwischen den verschiedenen Modellen macht Fehler meist offensichtlicher und macht damit das Verbessern der Modelle einfacher.

3.5.2.1 Metriken

Das definierte Modell könnte nun schon trainiert werden, jedoch sind oft zusätzliche Metriken während dem Training und dem Evaluieren des Modells wichtig, um festzustellen ob sich das Modell wie erhofft verhält. Einige nützlichsten Metriken sind folgende:

3.5.2.1.1 Confusion Matrix Eine Confusion Matrix bei binärer Klassifikation hat die Form $(2, 2)$ und beinhaltet 4 Werte:

- true positive gibt an, wie oft offene Fenster richtig vorhergesagt wurden
- true negative gibt an, wie oft geschlossene Fenster richtig vorhergesagt wurden
- false positive gibt an, wie oft bei einem geschlossenen Fenster als offen vorhergesagt wurde
- false negative gibt an, wie oft ein offenes Fenster als geschlossen vorhergesagt wurde

Diese Confusion Matrix an sich kann schon viel Einblick in das Verhalten des Modells geben, es können jedoch auch noch weitere Metriken daraus berechnet werden.

3.5.2.1.2 Accuracy gibt an, wie oft das Modell das Label richtig vorhersagt. Sie wird mit folgender Formel berechnet:

$$Accuracy = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Examples}} \quad (3.1)$$

3.5.2.1.3 Precision gibt an, wie oft das Modell richtig liegt, wenn es ein Fenster als offen vorhersagt. Wird berechnet durch:

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (3.2)$$

Precision ist eine wichtige Metrik, wenn die Kosten für falsche positive Vorhersagen sehr hoch sind.

3.5.2.1.4 Recall gibt an, wie viele offene Fenster das Modell richtig vorhergesagt hat. Wird berechnet durch:

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3.3)$$

Recall ist wichtig, wenn die Kosten für falsche negative Vorhersagen sehr hoch sind.

Auf ersten Blick wäre Accuracy eine gute Metrik für jedes Problem, jedoch hat es den Nachteil, dass es bei unbalanzierten Datensätzen, bei denen eine Klasse stark in der Mehrheit ist, nicht gut funktioniert. Es ist natürlich, dass Fenster öfter geschlossen sind als geöffnet, daher ist der gegebene Datensatz für dieses Problem unbalanciert. Im Extremfall könnte das Modell immer vorhersagen, dass das Fenster geschlossen ist und trotzdem eine Accuracy über 80% erzielen. Dies ist offensichtlich nicht das erwünschte Verhalten eines Modells.

Eine viel sinnvollere Metrik ist Recall. Die derzeitige Situation im Schulhaus ist, dass jeder Raum besucht werden muss, um alle Fenster zu schließen. Diese Anzahl an zu besuchenden Räumen soll verringert werden, während trotzdem noch so viele Fenster geschlossen werden wie möglich. Es sollte lieber ein extra Raum besucht werden müssen, in dem das Fenster geschlossen ist, als ein offenes Fenster zu übersehen. Das heißt, es sollen falsche negative Vorhersagen vermieden werden, was genau die Idee von Recall ist.

Es ist auch zu erwähnen, dass durch das Fokussieren auf eine Metrik andere Metriken schlechter ausfallen werden. Etwa bei Recall werden mehr positive Vorhersagen getroffen um einen größeren Anteil der positiven Fälle zu decken, wodurch mehr falsche positive Vorhersagen entstehen. Dies führt natürlich zu einer geringeren Accuracy und Precision. Daher ist die Entscheidung über welche Metrik genutzt werden soll eine extrem Wichtige.

3.5.3 Datenverwaltung

Bevor ein Modell trainiert werden kann wird natürlich ein Datensatz mit passendem Format benötigt.

3.5.3.1 Einlesen

Zuerst müssen alle verfügbaren Messpunkte aus der Datensammlung ausgelesen werden. Dies ist, wie bereits erwähnt, durch eine REST-Schnittstelle ermöglicht. Sie ist an der folgenden URL erreichbar:

<http://vm05.htl-leonding.ac.at/data/all>

An diesem Punkt sind die Daten im JSON-Format, welches nicht optimal zur Verarbeitung großer Mengen von Daten ist. Daher ist der nächste Schritt die wichtigen Daten, die Messpunkte, aus dem JSON-Objekt auszulesen und in einem Numpy Array zu speichern. Da jeder Messpunkt ein Merkmalsvektor der Länge 5 ist wird das Ergebnis ein 2-dimensionales Array, eine Matrix, der Größe $(N, 5)$ sein, wobei N die Anzahl der erhaltenen Messpunkte ist.

Bei diesem auslesen des JSON-Objektes ist jedoch Vorsicht geboten, da Sensoren und Messwerte fehlen können. Daher ist das endgültige N kleiner als die Anzahl der erhaltenen Messpunkte.

Das Label für jeden Messpunkt wird erhalten indem überprüft wird, ob die Anzahl der offenen Fenster, auch genannt "windowOpenCount" im JSON-Objekt, größer als 0 ist. Wenn True, dann ist das Label 1, andernfalls 0.

Nun muss diese Sequenz an Messpunkten umgewandelt werden zu einer Ansammlung von kleineren Sequenzen mit Länge 50 als bei der Definition des Modells besprochen. Dies wird durch Anwendung des "Sliding Window" Algorithmus erreicht, welcher wie folgt abläuft:

Es wird am Anfang der Sequenz an Messdaten $i = 1$ gestartet und ein "Window" der Länge 50 ausgelegt, bis zu $f = i + 50 - 1$. Alle Merkmalsvektoren $\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{f-1}, \mathbf{x}_f$ werden zu einer neuen Sequenz zusammengefasst und erhalten das Label y_f . Danach wird i , und daher auch f , um 1 erhöht und das nächste Sequenz-Label-Paar erzeugt. Dieser Ablauf wiederholt sich solange $f \leq N$.

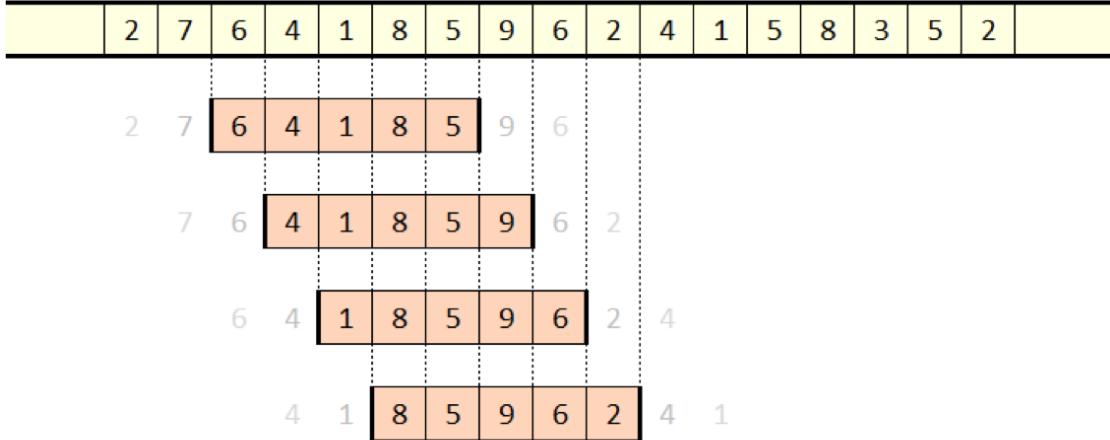


Abbildung 3.11: Beispiel für den "Sliding Window" Algorithmus anhand einer Sequenz von Skalaren.[8]

Die Ansammlung aller dieser Sequenz-Label-Paare $\left\{ \left(\{\mathbf{x}_{i+j}\}_{j=0}^{50-1}, y_{i+50-1} \right) \right\}_{i=1}^{N-50+1}$ ist der Datensatz für das Training des Modells und ist ein Tensor der Form $(N - 50 + 1, 50, 5)$.

Die Anwendung dieses Algorithmus garantiert eine maximale Anzahl an Trainingsdaten, was natürlich für das Modell von großem Vorteil ist.

In der Abbildung 3.12 ist eine Python Funktion zu sehen, welche den gesamten Ablauf zum Einlesen und Formatieren der Sensordaten beinhaltet. Diese Funktion nimmt als Parameter die Länge der geteilten Sequenzen, welche wie bisher besprochen 50 ist, die Namen der Sensoren im JSON-Objekt welche ausgelesen werden sollen und die Möglichkeit, ein Maximum für die ausgelesenen Messpunkte zu setzen, was nur für spätere Testzwecke sinnvoll ist. Zurückgegeben wird ein Tensor von Rang 3 und ein Vektor mit den dazugehörigen Labels.

3.5.3.2 Normalisieren

Wie bereits im theoretischen Teil besprochen sollten Inputs zu einem Bereich von beispielsweise $[-1; 1]$ skaliert werden. Um dies zu erreichen wurde folgender Ablauf gewählt:

```

def get_data(time_steps, sensors, num_datapoints=None):
    if num_datapoints == None:
        resp = requests.get('http://vm05.htl-leonding.ac.at/data/all')
    else:
        resp = requests.get('http://vm05.htl-leonding.ac.at/data/all?max={}'.format(num_datapoints))
    if resp.status_code != 200:
        raise ApiError('GET /data/all {}'.format(resp.status_code))
    size = len(resp.json()['object'])
    arr = np.empty([size, 5])
    y = np.empty(size)
    index = 0
    for item in resp.json()['object']:
        if all([sensor in item for sensor in sensors]) and 'windowOpenCount' in item:
            arr[index] = np.array([item[sensor] for sensor in sensors])
            y[index] = 1 if item['windowOpenCount'] > 0 else 0
            if not np.isnan(arr[index]).any():
                index += 1
    arr = arr[:index]
    y = y[:index]
    size = arr.shape[0]
    x = np.empty([size - time_steps + 1, time_steps, 5])
    for i in range(time_steps, size + 1):
        x[i - time_steps] = arr[i - time_steps:i]
    return (x, y[time_steps-1:])

```

Abbildung 3.12: Python Funktion zum Einlesen und Formatieren von Sensordaten

Zuerst wird für jede Art von Sensor das Minimum und das Maximum der möglichen Werte ermittelt. Die "Range" an möglichen Werten für jeden Sensor wird definiert als Maximum - Minimum. Alle gegebenen Sensorwerte werden nun um ihre minimalen Werte vermindert und mit 2 multipliziert, dadurch ist der Wertebereich $[0; 2 * Range]$. Die endgültigen Werte ergeben sich, indem diese Sensorwerte durch ihre "Range" dividiert werden und 1 abgezogen wird.

Diese Berechnungen sind in der Form einer Python Funktion in der Abbildung 3.13 zu sehen. Diese Funktion nimmt einen beliebigen Tensor mit Rang 3 als Input und gibt einen normalisierten Tensor gleichen Ranges zurück.

```

def scale_data(X):
    min_X = X.min(axis=1).min(axis=0).reshape((1, 1, -1))
    max_X = X.max(axis=1).max(axis=0).reshape((1, 1, -1))
    nom = (X-min_X)*2
    denom = max_X - min_X
    denom[denom==0] = 1
    return (-1 + nom/denom, min_X, max_X)

```

Abbildung 3.13: Python Funktion zur Skalierung von Tensoren 3. Ranges

Zudem werden die Minima und Maxima zurückgegeben, da diese bei Vorhersagen auf neuen Messpunkten wichtig sind, um diese auf konsistente Weise zu normalisieren. Würden diese Minima und Maxima für jeden Datensatz an Messpunkten neu berechnet werden, würde dies zu falschen Vorhersagen führen.

3.5.3.3 Aufteilen

Der letzte Schritt in der Datenverwaltung ist das Aufteilen des gegebenen Datensatzes in drei verschiedene Teile. Um den Grund für diese Teilung klarzustellen, hier eine Erklärung:

Angenommen es wird der gesamte Datensatz zum Training des Modells genutzt. Dies hat den Vorteil das alle vorhandenen Daten zum Training beitragen und das Modell daher mehr Informationen über die Problemstellung erhält. Der Nachteil ist jedoch, dass dieses Modell extrem gut bei Metriken abschneidet indem es sich beim Trainieren den Datensatz merkt, nicht jedoch eine robuste, generelle Regel zur Vorhersage der Label. Außerdem stehen keine weiteren Daten zur Verfügung die das Modell noch nicht gesehen hat, um es zu evaluieren.

Aus diesem Grund wird ein Teil des Datensatzes dem Modell in der Trainingsphase enthalten, das Testset, um es später damit zu evaluieren und mögliche Defekte im Modell zu entdecken.

Wird nun entdeckt, dass die gewählten Metriken bei der Evaluation keine zufriedenstellende Werte haben und wird daher das Modell geändert und neu trainiert, so lernt das Modell indirekt besser am Testset zu sein und das Generalisieren des Modells wird dadurch beeinträchtigt.

Die Lösung hierzu ist wiederum den Datensatz beim Training in zwei Teile zu spalten, das Trainingsset und das Validierungsset. Das Modell wird mit dem Trainingsset trainiert und während der Trainingsphase mit dem Validierungsset bewertet. Wenn das Modell zufriedenstellende Ergebnisse bei Trainingsset und Validierungsset liefert wird es anhand dem Testset evaluiert. Es sollten keine weiteren Änderungen gemacht werden, nachdem das Testset genutzt wurde, um nicht indirekt davon zu lernen. Diese finale Evaluation zeigt, welche Ergebnisse bei komplett neuen Daten zu erwarten ist.

Der am häufigsten benutzte Weg um einen Datensatz zu teilen ist es, zufällig Datenpunkte ohne "zurücklegen" auszuwählen und anhand einer Wahrscheinlichkeit, etwa 80% Trainingsdaten, entweder dem neuen Datensatz A oder B hinzuzufügen.

Dieser Schritt kann an einem der neuen Datensätze wiederholt werden um die drei geforderten Datensätze zu generieren.

Eine Erweiterung dieser Art die Daten zu teilen ist das so genannte "stratifizieren". Dabei wird der Datensatz anhand der Klassen geteilt und diese Teile werden anhand der zuvor beschriebenen Weise gespalten zu vier Datensätzen $A_{positiv}, B_{positiv}, A_{negativ}, B_{negativ}$. Diese werden zuletzt zusammengeführt zu A und B. Dadurch wird garantiert, dass Klassen gerecht in den Datensätzen verteilt sind und nicht zufälligerweise alle Datenpunkte für offene Fenster im Testset verloren gehen. Dies ist eine wichtige Methode zur Spaltung von Daten, wenn Klassen unbalanziert sind.

Leider können diese Methoden hier nicht genutzt werden, da die einzelnen Datenpunkte von einander teilweise abhängig sind. Dies stammt von dem Weg, in dem die lange Sequenz an Messpunkten zu kürzeren Sequenzen zum Trainieren geteilt wurde, dem "sliding Window" Algorithmus. Da bei jedem verschieben des "Windows" 49 Messpunkte in der gekürzten Sequenz gleich bleiben enthält ein Datenpunkt sehr viele Informationen über den nächsten Datenpunkt. Wenn diese nun zu den drei geforderten Sets geteilt werden, enthält jedes dieser Sets extrem viele Information über die Anderen, was das Aufteilen nutzlos macht. Overfitting würde hier nicht auffallen, da das Modell durch das Trainingsset genug Informationen über das Validierungsset und Testset um diese mit sehr guter Genauigkeit durch die gemerkten Datenpunkte zu erraten.

Die nächste Möglichkeit wäre, zwei Punkte i und f in der Sequenz zu wählen und die Sets wie folgt zu definieren:

$$\begin{aligned} \text{Trainingsset} &= \{(\mathbf{X}_j, y_j)\}_{j=1}^i \\ \text{Validierungsset} &= \{(\mathbf{X}_j, y_j)\}_{j=i+1}^f \\ \text{Testset} &= \{(\mathbf{X}_j, y_j)\}_{j=f+1}^N \end{aligned}$$

Dadurch würden nur die Informationen an den Schnittpunkten in mehreren Sets vorhanden sein, was durch die Menge der gesamten Datenpunkte ignoriert werden kann.

Das Problem hierbei ist, dass diese Abschnitte nicht repräsentativ für den gesamten Datensatz sind. Beispielsweise wurde im Sommer mit der Datensammlung begonnen und im Winter das Training des Modells gestartet. Bei einer Aufteilung mit zwei Punkten wie besprochen, würde das Modell auf Daten aus dem Sommer und frühen Herbst trainieren, aber mit Daten aus dem späten Herbst und Winter evaluiert werden. Im Winter sind nicht nur die Wetterverhältnisse substantiv

anders als im Sommer, sondern auch das Verhalten beim öffnen der Fenster ist anders, es wird weniger und kürzer gelüftet. Aus diesem Grund kann das Modell keine zufriedenstellende Leistung erbringen.

3.5.3.4 Umsetzung

Die letztendlich entwickelte Lösung zum Aufteilen des Datensatzes zu Trainingsset, Validierungsset und Testset ist folgende:

Es werden mehrere Punkte gewählt, welche den Datensatz in eine beliebige Anzahl kleinerer Abschnitte zerteilen. Diese Abschnitte werden nun mit 2 Schnittpunkten nach der gewünschten Verteilung in des Sets gespalten und den jeweiligen Sets hinzugefügt.

Nur dies ist jedoch noch nicht optimal, da neue Daten am Ende des Datensatzes immer im Validierungsset und Testset landen. Daher wird vor dem beschriebenen Ablauf der Datensatz in der Mitte gespalten und auf der ersten Hälfte normal angewandt. Bei der zweiten Hälfte wird die Anordnung der anteilhaften Zuweisungen zu den Sets umgedreht, so dass die neueren Datenpunkte im Trainingsset landen. Dadurch ist sichergestellt, dass die wichtigsten Zeiten der Sequenz, der Anfang im Sommer und das Ende im Winter, dem Modell zum Training zur Verfügung stehen.

Bei dieser Lösung ist jedoch das stratifizieren der Sets nicht möglich, daher sollte nach dem Aufteilen überprüft werden, ob alle Sets eine ähnliche Aufteilung der Klassen besitzen.

3.5.4 Training

Bevor das Modell letztendlich trainiert werden kann müssen einige Entscheidungen getroffen werden, die bestimmen wie das Modell trainieren wird.

Die ersten Entscheidungen betreffen den Optimierer welcher das Modell jede Epoche verbessert. Zuerst sollte geklärt werden, welcher Optimierer genutzt werden soll.

Zwei beliebte Optimierer sind Stochastischer Gradient Descent SGD und Adam. Während Stochastischer Gradient Descent garantiert das optimale Modell für das gegebene Trainingsset erlernt ist er um einiges langsamer als Adam.

Adam kann zwar in Ausnahmefällen die optimale Lösung verfehlten, dies ist in der Praxis jedoch extrem selten. Der Vorteil von Adam ist, dass Modelle sehr rapide trainiert werden können, was bei Prototyping sehr viel Zeit einspart. Aus diesem Grund wurde für diese Diplomarbeit der Adam Optimierer gewählt.

Zwei Parameter, die für den Optimierer gesetzt werden können, sind:

- die Lernrate, wie im theoretischen Teil besprochen
- der "Decay", welcher die Lernrate über den Lernvorgang hinweg verringert

Die nächste Entscheidung ist die "Batch size", also die Größe der Batches in die das Trainingsset jede Epoche geteilt wird. Vorteile einer kleinen Batch size:

- Benutzt weniger Speicher, was wichtig sein kann wenn das Trainingsset zu groß für den internen Speicher ist.
- Die Gewichte im Modell werden öfter aktualisiert, wodurch Konvergenz schneller erreicht wird.

Nachteil einer kleinen Batch size:

- Die berechneten Ableitungen der einzelnen Batches sind weniger repräsentativ für die Ableitung des ganzen Trainingssets, was schlechteren Updates für die Gewichte führt und daher Konvergenz langsamer macht oder sogar verhindert.
- Bei Problemen mit unbalanzierten Klassen kann es dazu kommen, dass ein Batch keine Instanz der Klasse in der Minderheit enthält, was zu schlechten Updates der Gewichte führt.

Wegen diesen Nachteilen wurde eine mittelmäßig bis große Batch size für dieses Problem gewählt.

Nun müssen Überlegungen zu den Epochen angestellt werden. Der erste Parameter der zu entscheiden ist, ist die maximale Anzahl an Epochen die Trainiert werden sollen. Diese hängt sehr von der Lernrate und der Komplexität des Modells ab.

Oft ist es jedoch nicht sinnvoll für die maximale Anzahl an Epochen zu trainieren, da schon vor dem Ende zu erkennen ist, dass das Modell anfängt zu overfitten oder sich nicht weiter verbessert. Meist lernt das Modell eine robuste Regel bis zu einem Punkt, an dem es übergeht zum Merken von Daten, overfitten.

Eine Lösung für dieses sinnlosen Training ist frühes Stoppen. Hierbei wird eine Metrik während dem Training beobachtet und wenn sich diese verschlechtert wird das Training gestoppt.

Da es jedoch vorkommen kann, dass das Modell kurz schlechter wird bevor es wieder besser wird, wird der Parameter "patience" eingeführt. Dieser Parameter gibt an, wie viele Epochen auf eine Verbesserung der Metrik über den besten gemessenen Wert gewartet wird. Wenn in diesen Epochen keine Verbesserung stattfindet werden die Gewichte zurückgesetzt auf die beste Instanz. Um overfitting entgegenzuwirken kann als Metrik auch als gemessen am Validierungsset benutzt werden, da nur generelle Verbesserungen zu Verbesserungen beim Validierungsset führen sollten.

Für die gegebene Problemstellung wurde als Metrik der Recall am Validierungsset gewählt, da ja Recall maximiert werden soll. Die maximalen Epochen wurden auf 50 gesetzt und patience auf 10, also nicht sehr aggressives frühes Stoppen.

Da die Klassen im gegebenen Datensatz so unbalanciert sind wurde zudem entschieden, die Klassen zu gewichten. Das Gewichten einer Klasse führt dazu, dass Fehler bei dieser Klasse mehr beziehungsweise weniger beim Berechnen der Kostenfunktion gewertet werden.

Normalerweise würde der Optimierer eher darauf achten so viele geschlossene Fenster wie möglich richtig zu vorhersagen, da diese wegen ihrer großen Anzahl im Datensatz eine gleich große Auswirkung auf die Kostenfunktion haben. Das Gewichten der Klassen wirkt dem entgegen, indem nun ein Fehler bei einem offenen Fenster mehr Kosten verursacht als bei einem geschlossenen.

Dieses Gewichten hat natürlich auch Auswirkungen auf die Metriken. Beispielsweise wird die Accuracy des Modells sinken, da nun Fehler bei der größeren Klasse in Kauf genommen werden um weniger Fehler bei der kleineren zu machen. Auf Recall wirkt es sich wiederum positiv aus, da nun mehr Fokus auf das erkennen offener Fenster gelegt wird. Durch die Gewichtung der Klassen kann dieses Verhältnis zwischen Recall und Accuracy beeinflusst werden, zu extreme Gewichte führen leicht zu unbrauchbaren Modellen.

```

neg, pos = np.bincount(Y.astype(int))
weight_for_neg = (1 / neg)*(Y.shape[0])/2.0
weight_for_pos = (1 / pos)*(Y.shape[0])/2.0

class_weight = {0: weight_for_neg, 1: weight_for_pos}
class_weight

{0: 0.630397196800295, 1: 2.417219128436312}

```

Abbildung 3.14: Python Code zur Berechnung der Gewichte der Klassen

Für das gegebene Problem wurde entschieden die Gewichte anhand der vorhandenen Anteile der Klassen im Datensatz zu berechnen, so dass es die gleichen Kosten verursacht alle geschlossenen Fenster falsch vorherzusagen als die offenen. Die Berechnung dafür wird in Abbildung 3.14 dargestellt.

Meist wäre eine weitere Entscheidung welche Loss Funktion zu nutzen. Da jedoch schon beim definieren des Modells klar war, dass die Problemstellung binäre Klassifizierung ist, bietet sich "binary crossentropy" perfekt an, da es dafür kreiert wurde.

Nachdem all diese Entscheidungen getroffen wurden muss all dies nur noch dem TensorFlow Framework übergeben werden, welches sich um das Training kümmert. Dies ist in der Abbildung 3.15 zu sehen. Diese Python Funktion erwartet ein Ansammlung von Name-Modell-Paaren als ersten Input. Dies sind alle Modelle die gleichzeitig trainiert werden sollen und deren Namen um sie zu identifizieren. Danach werden die besprochenen Parameter erwartet.

Da die Modelle per Referenz übergeben werden, müssen sie nach dem Trainieren nicht zurückgegeben werden, sie werden auch in der originalen Struktur verändert, nicht nur lokal. Der Rückgabewert ist eine Ansammlung von Name-Verlauf-Paaren, welche für jedes Modell dessen Verlauf der Metriken während dem Trainieren beinhaltet. Diese können zur Bewertung von Modellen genutzt werden, als im nächsten Abschnitt erklärt wird.

```

def train_models(model_dict, train, validation, learning_rate, decay,\n                max_epochs, batch_size, patience, class_weight=None):\n    metrics = get_metrics()\n    callbacks = get_callbacks(patience)\n    optimizer = get_optimizer(learning_rate, decay)\n\n    histories = {}\n\n    for (name, model) in model_dict.items():\n        model.compile(optimizer=optimizer,\n                      loss='binary_crossentropy',\n                      metrics=metrics)\n\n        model.summary()\n\n        histories[name] = model.fit(\n            train[0],\n            train[1],\n            epochs=max_epochs,\n            batch_size=batch_size,\n            validation_data=validation,\n            callbacks=callbacks,\n            class_weight=class_weight,\n            verbose=0)\n\n    return histories

```

Abbildung 3.15: Python Funktion zum Trainieren von Modellen

3.5.5 Bewertung

Das Bewerten von Modellen ist grob in zwei Abschnitte geteilt. Einerseits müssen die Modelle beobachtet und bewertet werden um Schlussfolgerungen auf mögliche Verbesserungen am Modell treffen zu können. Andererseits muss die Performance des finalen Modells evaluiert werden um das Verhalten bei neuen Daten zu zeigen.

3.5.5.1 Prototyping Phase

Während dem Entwickeln von Modellen werden deren Verhalten beim Training beobachtet um Probleme zu finden und diese zu beheben, damit die nächste Iteration an Modellen bessere Verhalten und daher auch bessere Metriken aufweist.

An diesem Punkt muss ein Modell oft nur in eine von zwei Kategorien eingeteilt werden, um zu wissen welche Schritte zur Verbesserung des Modells genommen werden können. Diese zwei Kategorien sind:

- Overfitting: das Modell merkt sich die Trainingsdaten und hat daher sehr gute Performance am Trainingsset, jedoch sehr schlechte am Validierungsset
- Underfitting: das Gegenteil von Overfitting. Das Modell ist nicht komplex genug um eine robuste, generelle Regel zu lernen. Zeigt sich durch generell schlechte Performance.

In der Abbildung 3.16 werden die Verläufe von vier Metriken bei einigen Modellen im Training gezeigt, mit einzelnen Linien für das Trainingsset und Validierungsset. Im Abschnitt 3.16a ist zu erkennen, dass alle Linien nahe aneinander sind aber keine guten Werte aufweisen, ein Anzeichen für Overfitting. Im Abschnitt 3.16b haben manche Linien gute Werte, die des Trainingssets, jedoch sind die Werte am Validierungsset um einiges schlechter, ein Anzeichen für Underfitting.

Beim Verbessern eines Modells ist die Regel es so lange komplexer zu machen, bis es overfittet, und dann dem Overfitting entgegenzuwirken. Dies soll zu einem Modell führen, welches so komplex wie nötig ist um optimale Ergebnisse zu erzielen.

Die Lösung zu Underfitting ist mehr Komplexität im Modell, das heißt mehr Layers, mehr Neuronen in den Layers, andere Arten von Layers und andere Aktivierungsfunktionen.

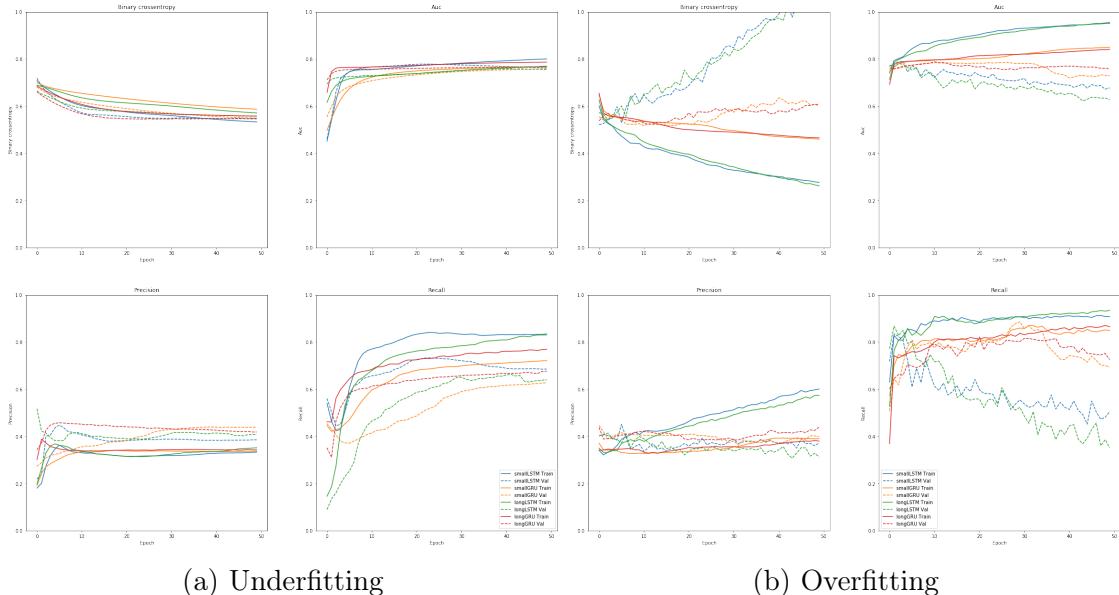


Abbildung 3.16: Verhalten von schlechten Modellen beim Training

Daraus folgt, dass die Lösung für Overfitting weniger Layers und weniger Neuronen in den Layers ist. Es gibt jedoch auch noch weitere Möglichkeiten um Overfitting zu verringern, beispielsweise "Dropout", wobei im Training zufällige Verbindungen zwischen einem Layer und dem Nächsten abgeschaltet werden, oder "L2 Normalisierung", wodurch die Größe der Gewichte im Modell zur Kostenfunktion beitragen.

Welche Schritte zur Verbesserung der Modelle gewählt werden hängt sehr stark mit der Erfahrung des Entwicklers zusammen und kann daher oft viele Iterationen von Trainieren und Verbessern benötigen bis ein akzeptables Modell entsteht.

3.5.5.2 Evaluation

Wenn ein Modell gefunden wurde, welches die Aufgabenstellung zu einem akzeptablen Grad erfüllt, wird es anhand des Testsets evaluiert. Die gemessene Performance kann im weiteren Verlauf für komplett neu gemessene Datenpunkte erwartet werden.

Es sollten keine weiteren Änderungen am Modell vorgenommen werden die auf der gemessenen Performance basieren, da ansonsten das Testset indirekt zum Trainieren des Modells genutzt wird.

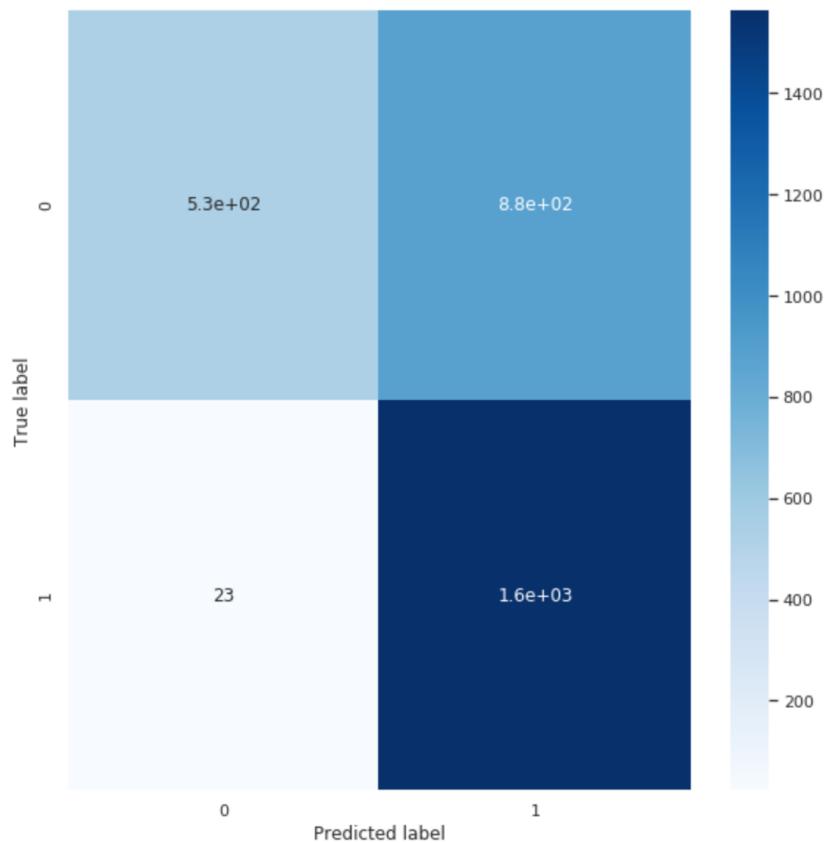


Abbildung 3.17: Confusion Matrix des trainierten Modells bei neuen Daten

Danach kann die Performance des Modells im praktischen Einsatz gemessen werden, indem auf das Messen neuer Datenpunkte gewartet wird und diese als neues Testset dienen.

Falls die Performance beim neuen Testset um einiges schlechter ist als beim originalen Testset, liegt dies entweder daran, dass die neuen Daten neue, unbekannte Verhalten abbilden (neue Wetterverhältnisse, ...) oder dass die Trainings-, Validierungs- und Testsets Informationen über einander enthalten und daher Overfitting verbergen.

Beim evaluieren des Modells an 3000 komplett neuen Datenpunkten wurde ein Recall von 98,5% erzielt. Die dazugehörige Confusion Matrix ist in der Abbildung 3.17 zu sehen. Wie wegen dem Fokus auf Recall erwartet ist zu erkennen, dass beinahe alle offenen Fenster erkannt werden, jedoch im Austausch dafür viele Fehler bei geschlossenen Fenstern gemacht werden.

3.5.6 Abspeichern

Das Modell selbst, also die definierte Struktur und alle gelernten Gewichte, kann von TensorFlow in verschiedenen Wegen gespeichert werden. Für dieses Projekt wurde der Export zu einem "Hierarchical Data Format"-File gewählt, da dies der Standard bei Keras Modellen ist.

Zusätzlich wird ein JSON-File erstellt, welches die Länge für Sequenzen beinhaltet, die das Modell akzeptiert, die Größe von Batches und die Maxima und Minima der gemessenen Sensorwerte, welche zum skalieren der Daten genutzt werden müssen.

Zum speichern von Modellen wurde eine Python Funktion geschrieben, welche eine gesamte Ansammlung von Name-Modell-Paaren, wie im Training genutzt, an einem gegebenen Pfad abspeichert. Das Ergebnis ist ein Ordner für jedes Modell, benannt nach dem Modell, welcher zwei Dateien enthält: model.h5 und config.json.

Solch ein Ordner kann später im MQTT Client der künstlichen Intelligenz geladen werden um automatisch Vorhersagen zu liefern.

3.6 MQTT Clients [CP]

Für das laufende System sind 2 MQTT Clients vorgesehen. Diese wurden in Python implementiert und laufen in Docker Containern.

3.6.1 Automatischer WakeUp

Dieser MQTT Client schickt in einem Intervall von 5 Minuten eine leere Nachricht an das Topic "htlleonding/predict_windows" um die künstliche Intelligenz "aufzuwecken".

Dieser Container wird automatisch im docker-compose des Gesamtsystems gestartet.

3.6.2 Künstliche Intelligenz

Dieser MQTT Client wartet auf Nachrichten zum Topic "htlleonding/predict_windows", trifft eine Vorhersage für den Status der Fenster und stellt diese über MQTT anderen Anwendungen zur Verfügung.

Zuerst wird ein Modell geladen, welches im vorhin besprochenen Format gespeichert wurde. Dieses Modell wird im Container am Pfad "/model" erwartet. Der einfachste Weg dies zu erfüllen ist es, das lokale Modell beim Starten des Container dort hin zu "mappen". Dazu kann Folgender Parameter bei "docker run" übergeben werden:

```
-v #lokaler Pfad#:model
```

Wobei #lokaler Pfad# mit dem Pfad zum Ordner des gespeicherten Modells ersetzt werden muss.

Wenn die Künstliche Intelligenz aufgeweckt wurde, wird ein Datensatz von der REST Schnittstelle der Datensammlung angefordert. Im Gegensatz zur Anfrage beim Training des Modells werden nun nur die n neuesten Messpunkte abgefragt, wobei n die Länge der benötigten Sequenzen aus der Datei config.json ist. Falls einer oder mehrere Messpunkte fehlerhaft sind wird keine Vorhersage getroffen.

Die Daten werden wie beim Training normalisiert, jedoch werden nun die Maxima und Minima aus der Datei config.json genutzt, welche aus dem Training des Modells stammen, anstatt sie neu zu berechnen. Würden die Maxima oder Minima neu berechnet werden, würde dies zu fehlerhaften Vorhersagen führen.

Nun kann das geladene Modell auf die normalisierten Daten angewendet werden um eine Vorhersage zu treffen. Diese Vorhersage ist jedoch noch immer eine Wahrscheinlichkeit, daher muss geprüft werden ob sie über 50% liegt um ein binäres Ergebnis zu erhalten.

Zuletzt wird dieses Ergebnis an das Topic "htlleonding/firstfloor/e581/pc/window/state" geschickt. Eine solche Nachricht besteht aus:

- Timestamp: dem Zeitpunkt an dem die Vorhersage getroffen wurde
- Value: der Status der Fenster. Entweder "Open" oder "Closed"

Kapitel 4

Fazit und Ausblick

Ein bereits sehr komplexes System wurde überarbeitet und weiterentwickelt. Da derzeit nur in einem Raum Sensordaten gesammelt werden und diese Daten nicht in der bereits bestehenden Datenbank gespeichert werden und daher auch nicht über die dazu gehörige REST-Schnittstelle zugänglich sind, wurde die künstliche Intelligenz sehr limitiert. Sie ist nun auf die neue Datensammlung angewiesen und kann nur für einen Raum Vorhersagen treffen. Des Weiteren konnte nicht überprüft werden, ob das trainierte Modell vergleichbare Performance bei neuen Räumen aufweist. Eine höhere Anzahl an Messdaten, beides durch längere Datensammlung und durch Erweiterung zu neuen Räumen, würde außerdem zu besseren Ergebnissen beim Trainieren der künstlichen Intelligenz führen.

Der Ausbau der Datensammlung würde jedoch nicht nur für die künstlichen Intelligenz Vorteile bringen, sondern auch dem gesamten Rest des Systems. Zur Zeit hat das System keine Versorgung von echten Messwerten und wird dadurch äußerst in seiner Nützlichkeit eingeschränkt.

Das Videostreamen wurde dabei nicht dauerhaft installiert. Weiters könnte der Videostream um Ton erweitert werden.

Für das 3d - Model ist es noch möglich eine Steuerung zu entwickelt. Diese Steuerung könnte dann Beispielweise vom Leonie - Projekt verwendet werden. Eine eingebaute Navigation und Raumsuche könnte die Orientierung im Schulgebäude erleichtern.

Mit den in dieser Diplomarbeit vorgenommenen Erweiterungen ist das Potential dieses Projektes noch nicht Ansatzweise ausgeschöpft. Der Aufbau des Systems eignet sich besonders gut Komponenten hinzuzufügen oder auszutauschen, daher wird dieses Projekt sicherlich zu weitere Diplomarbeiten an der HTL - Leonding führen. Wenn es letztendlich in allen Klassenräumen integriert ist, wird das Projekt IoT - Smart School große Auswirkungen an der Schule haben, sei es die Verbesserung der Lebensqualität der Schüler in den Klassenräumen oder das Einsparen von unnötigen Kosten durch bessere Überwachung.

Literatur

- [1] *Explaining Recurrent Neural Networks*. Bouvet Norge. Library Catalog: www.bouvet.no. URL: <https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks> (besucht am 11.03.2020).
- [2] *Fig. 1. Artificial neural network architecture (ANN i-h 1-h 2-h n-o)*. ResearchGate. Library Catalog: www.researchgate.net. URL: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051 (besucht am 09.03.2020).
- [3] *Fig. 11. Various forms of non-linear activation functions (Figure...)*. ResearchGate. Library Catalog: www.researchgate.net. URL: https://www.researchgate.net/figure/Various-forms-of-non-linear-activation-functions-Figure-adopted-from-Caffe-Tutorial_fig3_315667264 (besucht am 09.03.2020).
- [4] *Git Branches Free Tutorial*. en. Library Catalog: www.nobledesktop.com. URL: <https://www.nobledesktop.com/learn/git/git-branches> (besucht am 03.04.2020).
- [5] *Gradient Descent Visualization - File Exchange - MATLAB Central*. Library Catalog: uk.mathworks.com. URL: <https://uk.mathworks.com/matlabcentral/fileexchange/35389> (besucht am 09.03.2020).
- [6] Tom M Mitchell u. a. *Machine learning*. 1997.
- [7] *PI4 MODEL B/4GB & POWER SUPPLY & SD CARD Raspberry Pi 4 15 GHz QuadCore und RNDNetzteilPaket, 4GB RAM Raspberry Pi*. de. Library Catalog: www.distrelec.at. URL: <https://www.distrelec.at/de/raspberry-pi-ghz-quad-core-und-rnd-netzteil-paket-4gb-ram-raspberry-pi-pi4-model-4gb-power-supply-sd-card/p/30158761> (besucht am 03.04.2020).
- [8] *Printing lines with sliding windows in bash*. Library Catalog: zhiganglu.com. 1. Juli 2019. URL: <https://zhiganglu.com/post/bash-print-sliding-windows/> (besucht am 02.04.2020).

- [9] Sebastian Raschka und Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition*. Packt Publishing, 12. Dez. 2019. 770 S. ISBN: 978-1-78995-575-0.
- [10] *Reprint: A text to understand the back propagation method in neural networks - BackPropagation - Programmer Sought*. URL: <http://www.programmersought.com/article/8371102793/> (besucht am 03.04.2020).
- [11] Arthur L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM J. Res. Dev.* (1959). DOI: 10.1147/rd.33.0210.
- [12] *Sigmoid function*. In: *Wikipedia*. Page Version ID: 944113373. 5. März 2020. URL: https://en.wikipedia.org/w/index.php?title=Sigmoid_function&oldid=944113373 (besucht am 09.03.2020).
- [13] techeplanet. *JSON Example - A JSON Example That Includes Complex Data Structure*. Library Catalog: techeplanet.com Section: JSON. 4. Jan. 2019. URL: <https://techeplanet.com/json-example/> (besucht am 11.03.2020).
- [14] *Understanding WebRTC Media Connections — ICE, STUN, and TURN*. en. Library Catalog: andrewjprokop.wordpress.com. Juli 2014. URL: <https://andrewjprokop.wordpress.com/2014/07/21/understanding-webrtc-media-connections-ice-stun-and-turn/> (besucht am 02.04.2020).
- [15] *What is MQTT and How It Works*. en-US. Library Catalog: randomnerdtutorials.com. Sep. 2018. URL: <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/> (besucht am 03.04.2020).

Abbildungsverzeichnis

2.1	Erklärung von Stun & Turn [14]	16
2.2	Raspberry Pi 4 [7]	17
2.3	Mqtt [15]	29
2.4	Git - Visuell [4]	33
2.5	Illustration des Ablaufes für Supervised Learning.[9]	40
2.6	Illustration des Ablaufes für Reinforcement Learning.[9]	41
2.7	Lineare Regression für ein-dimensionale Beispiele.[9]	43
2.8	Darstellung der Kostenfunktion bei ein-dimensionaler Linearer Regression und dem Verlauf von Gradient Descent.[5]	45
2.9	Darstellung der Sigmoid Funktion.[12]	46
2.10	Darstellung der Trennung von Klassen durch ein gelerntes Modell (strichlierte Linie).[9]	47
2.11	Illustration eines Neuronalen Netzwerkes.[2]	48
2.12	Illustration von Forward Feeding.[10]	49
2.13	Illustration der wichtigsten Aktivierungsfunktionen.[3]	51
2.14	Illustration eines Recurrent Neural Networks.[1]	52
2.15	Beispiel für eine JSON Datei.[13]	58

3.1	Architekur der Gesamtsystem. Der Raspberry Pi, Das 3d - Model welches ein Teil des IoT - Web Teil ist und die Künstliche Intelligenz, alle in Blau abgebildet, wurden in dieser Diplorarbeit entwickelt.	59
3.2	3d - Model mit aktiven Raum	70
3.3	MQTT - HTML Code	70
3.4	MQTT - Typescript Code	71
3.5	3d - Model mit aktivierter Filter	72
3.6	3d - Model mit aktivierter Filter	73
3.7	Illustration der System Architektur	79
3.8	Ein Lupyter Lab Notebook	80
3.9	Beispiel für http://vm05.hlt-leonding.ac.at/data/all?max=2	81
3.10	Die ersten definierten Modelle	85
3.11	Beispiel für den "Sliding Window" Algorithmus anhand einer Sequenz von Skalaren.[8]	89
3.12	Python Funktion zum Einlesen und Formatieren von Sensordaten .	90
3.13	Python Funktion zur Skalierung von Tensoren 3. Ranges	90
3.14	Python Code zur Berechnung der Gewichte der Klassen	96
3.15	Python Funktion zum Trainieren von Modellen	97
3.16	Verhalten von schlechten Modellen beim Training	99
3.17	Cunfusion Matrix des trainierten Modells bei neuen Daten	100