Technischer Aufbau

# Verwendete Technik

Für die Umsetzung des Projekts wurden die herkömmlichen Webtechnologien HTML, CSS und JavaScript verwendet. Zusätzlich wurde ein Backend mit PHP umgesetzt, welches wiederum mit einer MySQL-Datenbank zusammenarbeitet. Das Projekt wurde ohne spezielles JavaScript Framework realisiert. An einigen Punkte kamen jedoch Elemente von JQuery und dem damit verbundenen JQuery-UI zur Verwendung. Die Machine-Learning Modelle wurden in Python erstellt und trainiert.

# Umsetzung

Dieser Abschnitt geht etwas detaillierter auf die Umsetzung verschiedener Aspekte des Projekts ein. Allerdings wurde bei weitem nicht jede Funktion bzw. jede technische Umsetzung beschrieben, da dies den Rahmen der Dokumentation übersteigen würde. Stattdessen wurde versucht, ein Überblick über die wichtigsten und markantesten Punkte der technischen Umsetzung zu geben.

# Machine-Learning

Wie bereits erwähnt, wurden alle Modelle mithilfe von Python erstellt und trainiert. Als Machine-Learning Technologie kam Tensorflow und das darauf aufbauende Keras zum Einsatz. Die Modelle der verschiedenen Kategorien sind Großteiles gleich aufgebaut, benötigten aber an einigen Stellen auch individuelle Anpassungen. Die verschiedenen Hyperparameter wurden im Laufe der Entwicklung für die Modelle immer wieder angepasst und so verändert, dass möglichst gute Ergebnisse erzielt werden konnten. Das Modell, welches handgeschriebene Zahlen erkennen kann (learningDigitsOwnNumbers2.py), wurde noch zusätzlich um eine Funktion erweitert. Dieses Modell verwendet neben dem „MNIST“- Datensatz von Tensorflow auch die durch die Anwendung generierten Bilder für das Training. Die hierfür zuständige Funktion im Modell heißt „load\_images\_to\_data“. Um dies möglich zu machen, waren auch gewisse Änderungen im Bereich der Webanwendung anzupassen. Beispielsweise zu nennen wäre das Labeln der gespeicherten Bilder, für eine spätere korrekte Zuordnung.

# Machine-Learning im Frontend

Für das Verwenden der Modelle in der Webanwendung wurde Tensorflow.js verwendet. Mit dieser Bibliothek ist es möglich eigene Modelle zu trainieren oder auch einfach nur zu laden. Da die Modelle für dieses Projekt in Python trainiert wurden, mussten diese nur noch geladen werden. Nach dem Laden des entsprechenden Modells, kann ein Tensor an das Modell übergeben werden, worauf eine Vorhersage getroffen werden soll. Somit braucht es nur drei Schritte für das Vorhersagen einer Zeichnung. Laden des Modells, umwandeln der Zeichnung in einen Tensor und anschließend das Durchführen der Vorhersage. Das von der Tensorflow.js Funktion „predict“ zurückgegebene Ergebnis für eine Vorhersage sieht z.B. wie folgt aus.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung : Ergebnis Tensorflow.js Vorhersage

In diesem Beispiel wurde eine Vorhersage für die Zahlen 0 bis 9 gemacht. In diesem Fall wurde die Zahl 0 mit einer Wahrscheinlichkeit von 99,9% erkannt.  
Für die endgültige Auswertung und Überprüfung der Werte, welche das Modell zurück liefert, mussten für die unterschiedlichen Kategorien JavaScript Funktionen erstellt werden, da sich die Zuordnung der Ergebnisse nur bei den Zahlen so leicht gestaltet. Bei den Buchstaben z.B. werden 26 Indizes zurückgegeben, welche danach den Buchstaben des Alphabets zugeordnet werden müssen.

# Account-System

Die Gesamte Anwendung baut auf einem Account-System auf. Das heißt bevor ein Nutzer die Anwendung verwenden kann, muss ein Account erstellt werden und sich anschließend eingeloggt werden. Durch das Einloggen wird mithilfe von PHP eine Session gesetzt, durch welche zugeordnet werden kann, welcher Nutzer aktuell eingeloggt ist. Durch das Verwenden eines solchen Account-Systems, wird es möglich, jedem Nutzer seine individuellen Lernpläne und Lernerfolge zuzuordnen. All diese Informationen, werden in einer MySQL-Datenbank gespeichert. Die folgende Abbildung zeigt die Datenbankstruktur mit ihren Feldern und Beziehungen.

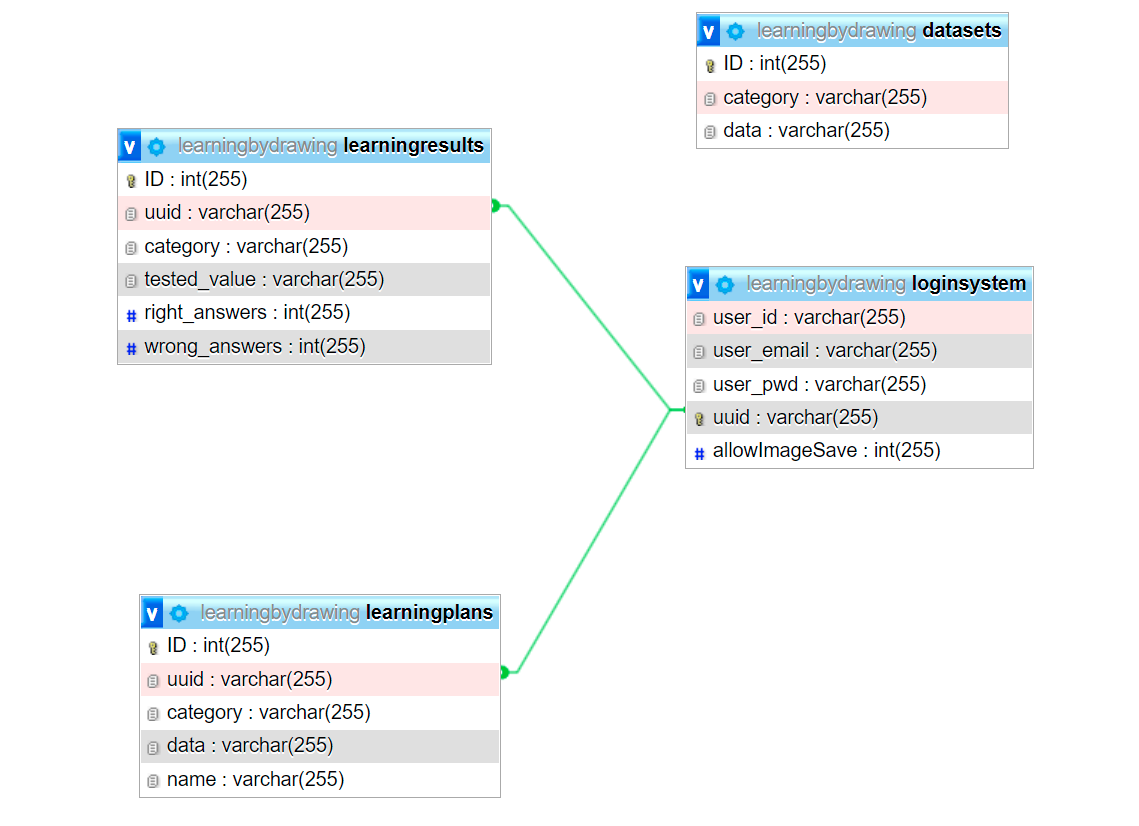


Abbildung : Datenbank Struktur und Beziehungen

Die Tabelle „datasets“ enthält die verschiedenen Kategorien mit den dazugehörigen Daten. Soll der Anwendung eine neue Kategorie hinzugefügt werden, muss einfach ein weitere Datensatz in die Datenbank eingepflegt werden. In der „loginsystem“ Tabelle werden alle direkten Informationen über den Nutzer gespeichert. Also z.B. der Benutzername, die E-Mail oder auch das Passwort. Das Passwort ist nicht im Klartext gespeichert, sondern wurde zuvor mit einem Salted Hash Algorithmus verschlüsselt. Außerdem wird in dieser Tabelle für jeden Nutzer festgehalten, ob das Speichern der gezeichneten Bilder gestattet wurde oder nicht. Die Tabelle „learningresults“ dient zum Speichern der Lernerfolge. Für jede Übung wird die Anzahl der richtigen und falschen Lösungen festgehalten. Über den Fremdschlüssel „uuid“, kann das jeweilige Ergebnis dem richtigen Nutzer zugeordnet werden. Die „uuid“ ist eine Eindeutige ID, welche jeder Nutzer bei der Erstellung des Accounts zugeordnet bekommt. In der Tabelle „loginsystems“ dient das Feld „uuid“ also als eindeutiger primär Schlüssel, auf welchen von den anderen Tabellen mit Fremdschlüsseln referenziert werden kann. In der Tabelle „learningplans“ werden die vom Nutzer erstellten Lernpläne gespeichert. Auch hier erfolgt die Zuordnung zum Nutzer über den Fremdschlüssel „uuid“.

# Backend

Im Backend (backend.php) werden sämtliche Datenbank Transaktionen gehandhabt. Mithilfe von If-Abfragen wird überprüft, welche Transaktion durchgeführt werden soll. Hierbei ist zu erwähnen, dass im Backend auch verschiedene Sicherheitsvorkehrungen getroffen wurden, um die Datenbank vor möglichen Angriffen oder nicht vorgesehenen Nutzeraktionen zu schützen. So wird zum Beispiel sichergestellt, dass die Werte, die vom Frontend an das Backend übergeben werden, alle valide bzw. zulässige Werte sind. Nicht zulässige Werte, die z.B. durch eine DOM-Manipulation erzeugt wurden, werden verworfen und nicht gespeichert. Außerdem sind alle Datenbank Transaktionen mit „Prepared-Statements“ umgesetzt. Das Datenbank System erhält die SQL-Statements dadurch ohne Parameter bzw. mit Platzhaltern für die Parameter. Erst nachdem die Parameter vom Datenbank System auf ihre Gültigkeit überprüft wurden, wird die Transaktion ausgeführt. Durch diese Methodik wird die Datenbank vor „SQL-Injections“ geschützt. Neben den Datenbank Transaktionen ist auch das Speichern der gezeichneten Bilder im Backend umgesetzt. Jedes Bild erhält das entsprechende Label für die spätere Zuordnung beim Machine-Learning, ein Datumsstempel und eine eindeutige ID als Name. Außerdem wird sichergestellt, dass der Nutzer die Erlaubnis erteilt hat, dass die Bilder gespeichert werden dürfen. Sollte diese Erlaubnis nicht erteilt sein, werden die Bilder nicht gespeichert.

# Machine-Learning Bildverarbeitung und Resemble.js

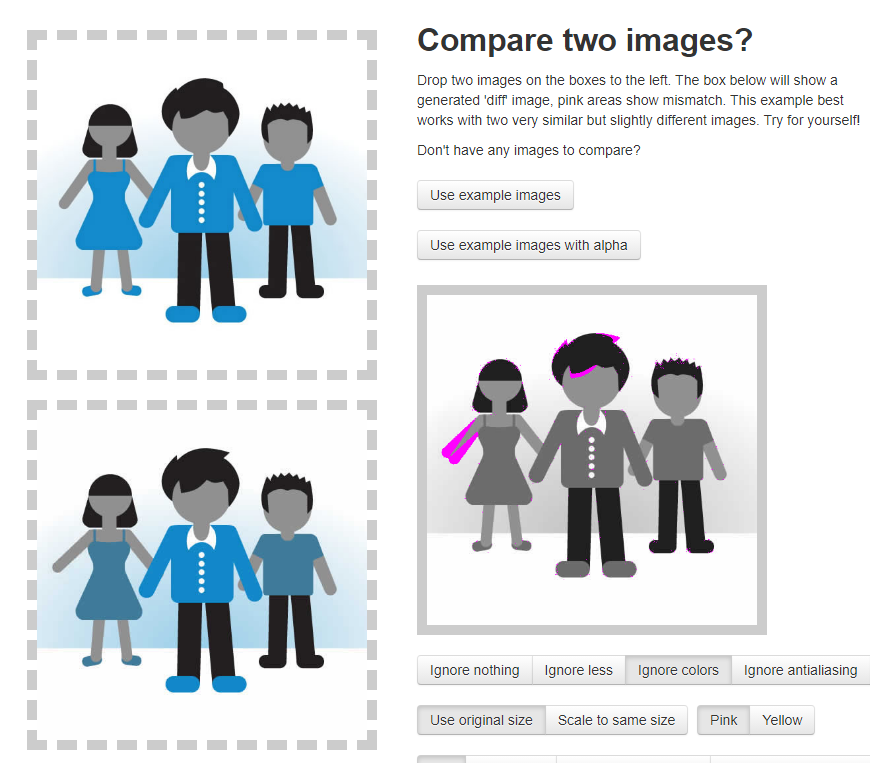
Die Erkennung der Freihandzeichnungen erfolgt mittels Bildverarbeitung durch verschiedene Machine-Learning Modelle bzw. neuronale Netze. Die Umsetzung dieses Features mit einem neuronalen Netz war auch schon in der Konzeptphase des Projekts so geplant. Allerdings kam auch ein Bildvergleich, zwischen optimaler Lösung und erstellter Nutzerlösung, in Frage. Im Rahmen der Recherche einer solchen Lösung tat sich Resemble.js für die Umsetzung eines solchen Bildvergleichs hervor. Nach näherer Untersuchung dieser Bibliothek wurde allerdings klar, dass schon kleinste Unterschiede in den Bildern zu einer hohen prozentualen Differenz in der Auswertung führen. Auf der Resemble.js Webseite wird ein Beispiel gezeigt, was dies auch nochmal verdeutlicht. 

Abbildung : Resemble.js Bildvergleich Beispiel

Die zwei Links angeordneten Bilder sind die Bilder, die verglichen werden sollen. Das Bild auf der rechten Seite stellt das Ergebnis dar. Im Ergebnisbild sind alle unterschiede pink markiert. Wie zu sehen ist werden die Bilder pixelgenau verglichen, was bei einer Freihandzeichnung wie in unserem Fall zu schlechten Ergebnissen führen würde. Außerdem wird für die verwenden von Resemble.js die JavaScript Laufzeitumgebung Node.js benötigt, was in diesem Projekt nicht vorgesehen ist. Aus diesen zwei genannten Gründen kam die verwendung eines Bildvergleichs bzw. die Verwendung von Resemble.js für dieses Projekt nicht in Frage.

# Probleme und Herausforderungen:

In diesem Abschnitt wird auf Probleme und Herausforderungen eingegangen, die während der Entwicklung der Anwendung aufgetreten sind.

## Daten / Datensätze für Machine Learning

Um Neuronale Netze trainieren zu können wird eine große Menge an relevanten Daten benötigt. Zusätzlich ist es in unserer Anwendung der Fall, dass das Neuronale Netz handgezeichnete Bilder von Zahlen, Buchstaben etc. erkennen muss. Somit ist ein weiterer wichtiger Punkt das diese Trainings Daten ebenfalls handgeschriebene Bilder enthalten sollen damit das Netzt auch zuversichtliche Vorhersagen treffen kann. Hier haben wir besonders im Bereich von geometrischen Formen keine richtig verlässlichen Daten gefunden. Wir mussten daher mit “fast“ perfekten Formen arbeiten. Das führt zu dem Problem das handgeschriebene Bilder schwieriger für das Netz einzuordnen sind. Auch bei der Anzahl von Daten gibt es sehr starke Abweichungen. Zahlen und Buchstaben sind zwar weit verbreitet allerdings gibt es zu komplexeren Daten / Formen / Zeichen etc. deutlich weniger Daten (freie Daten). Da jeder Mensch eine andere Handschrift hat ist es jedoch wichtig möglichst viele Daten (mehrere Tausend pro Klasse) zu besitzen. Auch sind generell Daten für manche Kategorien nicht frei zugänglich, nicht vorhanden oder qualitativ unpraktisch.

## Daten Vorbereitung

Auch die Vorbereitung der Daten war eine Herausforderung. Für ein neuronales Netz ist dabei z.B. auch Farbe entscheidend, z.B. oder der Hintergrund schwarz oder weiß ist. Da wir für die Anwendung einen Schwarzen Hintergrund nutzen, müssen die Trainings Daten am besten auch mit diesem Hintergrund von Bildern arbeiten. Allerdings sind daten dahingehen unterschiedlich und nicht genormed. Eine Herausforderung war hierbei die Daten so zu bearbeiten das sie möglichst gut für unseren Anwendungszweck geeignet sind.

Auch kommen Daten in anderen Formaten und Formen und wir mussten für jeden Datensatz bzw. Format darauf achten, dass wir die Daten anders importieren, lesen und vorbereiten. Z.B. auch Daten für Training Test und Validierung Trennen.

## Modelle und Training

Auch musste für jede Kategorie ein eigenes Neuronales Netz trainiert bzw. ein neues Modell erstellt werden. Dies hat viel Zeit in Anspruch genommen und erfordertet viel Code. Wir konnten auch nicht ein Modell für alle Netzt nutzen da dieses nicht auf allen Daten gleich funktioniert. Es wurde Basismodell entwickelt und dieses dann weiter auf die verschiedenen Daten angepasst, indem Layer hinzugefügt, entfernt und Hyperparameter angepasst wurden.

Ein Problem was unterschätzt wurde ist die Zeit und Rechenleistung, die das Trainieren einiger Modelle in Anspruch genommen hat. Da Hyperparameter hauptsächlich durch “Trial-and-Error“ angepasst werden mussten wir sehr viele Trainingstests durchlaufen was sehr Zeitintensiv ist. Je nach Komplexität des Netzes und der anpassbaren Parameter sowie Anzahl an Trainingsdaten kann ein Durchlauf von ca. 1 – 5 Stunden dauern. Auch wurden wir auf die Rechenleistung unserer Rechner beschränkt, was nur eine gewisse Komplexität der Modelle ermöglicht. Das führt dazu das in manchen Szenarien die Netze nicht immer eine Richtige vorhersage liefern und teils genauer sein könnten. Besonders im Bereich der geometrischen Formen ist das Netz hinsichtlich der Genauigkeit noch nicht perfekt, und etwas ungenau (hier spielen jedoch auch die Trainingsdaten eine Rolle).

Ausblick:

In diesem Abschnitt werden zukünftige Verbesserungen, Features/Funktionen und mögliche Technologie Integrationen vorstellt. Nachdem Funktionen definiert worden sind, haben sich jedoch während der Umsetzung oder beim Testen der Anwendung weitere Ideen gebildet die später umgesetzt und integriert werden könnten. Diese Inhalte und Funktionen werden nun aufgelistet.

## Weitere Lerninhalte:

Die Anwendung wurde so aufgebaut das sich recht einfach neue Lerninhalte hinzufügen lassen oder vorhandene erweitert werden können. So können komplexere Lerninhalte wie chinesische Hànzì oder alle japanischen Hiragana Zeichen hinzugefügt werden (aktuell nur 10 von 49). Auch komplexere geometrische Formen können hinzugefügt werden, um Quizfragen interessanter zu gestalten.

## Zukünftige Funktionen:

Funktionen, die nun aufgelistet werden, wurden nicht in der Grundidee bzw. dem Konzept beachtet und wurden deshalb hintenangestellt. Weiterhin haben einige dieser Funktionen nicht in den gegebenen Zeitraum des Projekts gepasst, können allerdings trotzdem sehr sinnvoll sein.

### Musterlösung

Aktuell erhält der Nutzer eine Rückmeldung von dem System, ob seine Antwort richtig war oder nicht. Als “Quality of Life“ Funktion könnte nach der Rückgabe des Systems eine Art Musterbild angezeigt werden. Dieses Musterbild repräsentiert hier dann die optimale Lösung. Der Nutzer wüsste was diese gewesen wäre und was er falsch gezeichnet hat.

### Detaillierte Konfiguration von Lernplänen

Eine weitere Funktion, die hinzugefügt werden könnte, ist eine Erweiterung von Lernplänen. Allgemein geht es hierbei um mehr Konfigurationsmöglichkeiten von Lernplänen. Man könnte Lernpläne aus mehreren Kategorien erstellt oder die Häufigkeit von einzelnen Quizfragen anpassen.

### Datensatz Erstellung von vertrauenswürdigen Personen

Wie in Probleme & Herausforderungen erwähnt wurde ist es nicht einfach Datensätze zu finden. Für diese Funktion kann man vertrauenswürdigen Personen die Möglichkeit geben bei der Erstellung von Datensätzen mitzuwirken. Dabei würden diesen Personen dann Musterlösungen vorgegeben, die sie dann nachzeichnen sollen. Diese Zeichnungen werden dann gelabelt, gespeichert und könnten dann zum Trainieren neuer Modelle genutzt werden.

### Manuelle Überprüfung

Ein Neuronales Netz ist nicht fehlerfrei und es kann durchaus vorkommen, dass eine Falsche Entscheidung getroffen. Hier könnte eine zusätzliche Einstellung eingefügt werden, die es Nutzer ermöglicht nach der Überprüfung seine eigene Antwort noch einmal zu Werten. Wird die Antwort trotzt der Entscheidung des Netzes als richtig eingestuft kann dieses Bild dazu genutzt werden das Modell weiter zu trainieren und zu verbessern, um solche Fehler zu verhindern.

## Weitere Technologien

Wie vorher schon beschrieben wurde ist ein klassischer Bildvergleich nicht sehr praktikabel für diese Anwendung. Jedoch kann man gewisse Vorverarbeitung eines eingehenden Bildes umsetzen und auch versuchen den klassischen Bildvergleich “gütiger“ zu werten bzw. zu gestalten sodass minimale Abweichungen oder andere Positionen nicht so stark gewichtet werden. Sinnvoll ist diese Technologie dahingehen, wenn es keine oder nur wenige Datensätze für Machine Learning gibt.