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 bzw. neuronalen Netze wurden in Python erstellt und trainiert.

Umsetzung

Dieser Abschnitt geht 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. In der „README“-Datei ist eine Erklärung zu finden, welche Teile der Anwendung in den Unterordnern des Projekts zu finden sind.

Machine-Learning

Wie bereits erwähnt, wurden alle neuronalen Netze 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 großteils 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. Die Links zu den verwendeten Trainings- und Testdaten sind in der

„README“ Datei des Projects aufgelistet. Jeder Python Datei des Projekts wurde die passende Quelle zugeordnet, von welcher die Daten für den Trainingsprozess stammen. Außerdem sind in den Python-Dateien an wichtigen Stellen erklärende Kommentare zu den durchgeführten Aktionen zu finden.

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.

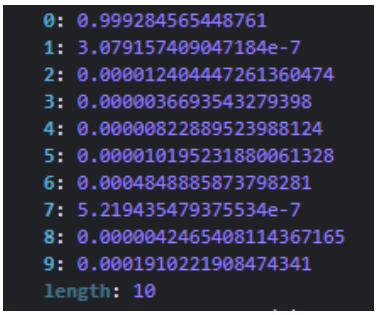

Abbildung 1: Ergebnis Tensorflow.js Vorhersage

In diesem Beispiel wurde eine Vorhersage für die Zahlen 0 bis 9 gemacht, wobei die Zahl 0 mit einer Wahrscheinlichkeit von 99,92% erkannt wurde.

Für die endgültige Auswertung und Überprüfung der Werte, welche das Modell zurückliefert, 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 werden z.B. 26 Indizes mit jeweils einer Wahrscheinlichkeit 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.

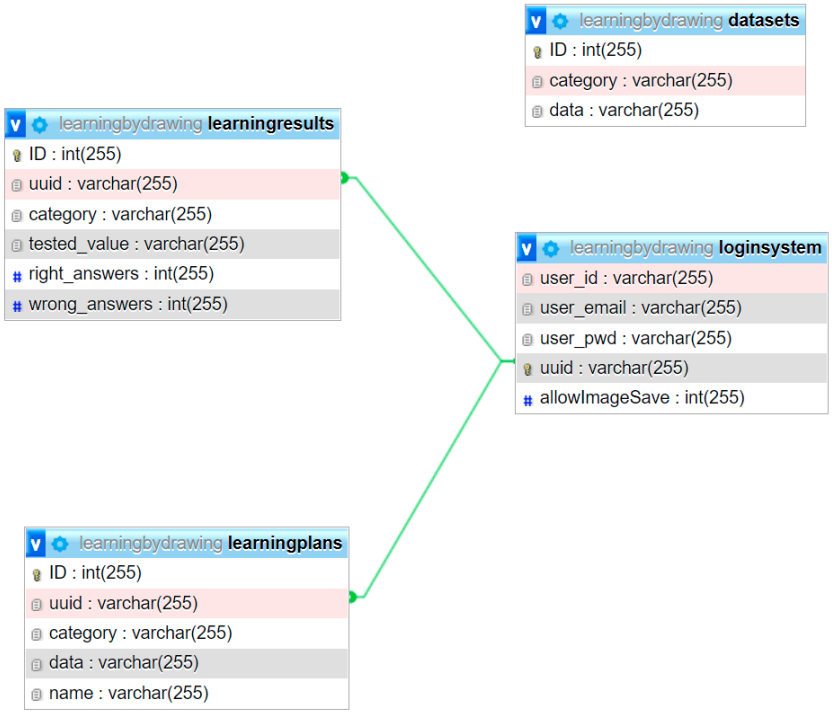

Abbildung 2: Datenbank Struktur und Beziehungen

Die Tabelle „datasets“ enthält die verschiedenen Kategorien mit den dazugehörigen Daten. Soll der Anwendung eine neue Kategorie hinzugefügt werden, muss einfach ein weiterer 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. Das

Feld „uuid“ enthält eine Eindeutige ID, welche jeder Nutzer bei der Erstellung des Accounts zugeordnet bekommt. In der Tabelle „loginsystems“ dient das Feld „uuid“ also als eindeutiger Primärschlüssel, auf welchen von den anderen Tabellen mit Fremdschlüsseln referenziert werden kann. Durch diese Primär- und Fremdschlüssel Beziehungen wird die referenzielle Integrität und die Konsistenz der Datenbank gewahrt. 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.

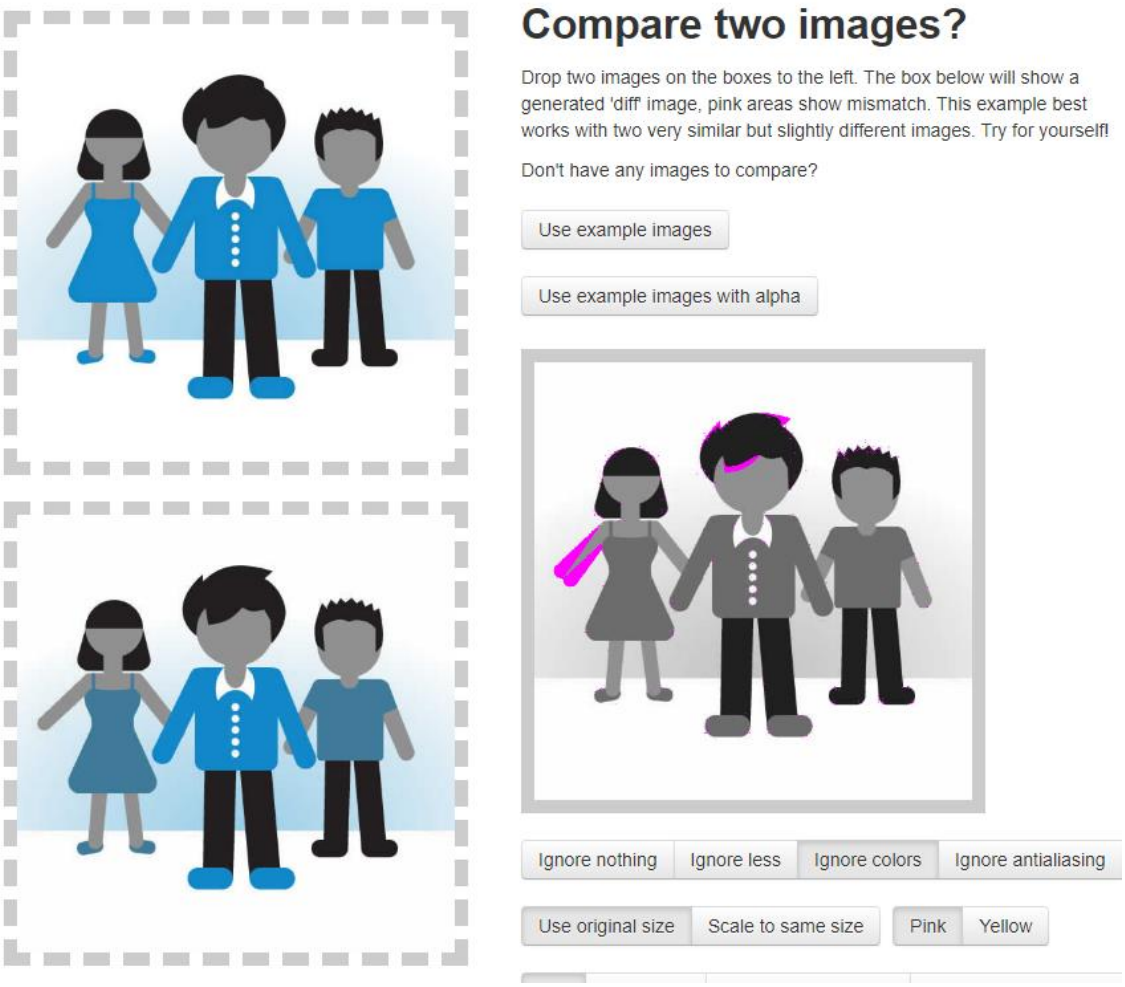

Abbildung 3: Resemble.js Bildvergleich Beispiel

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 Verwendung 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 der Anwendung der Fall, dass das neuronale Netz handgezeichnete Bilder von Zahlen, Buchstaben etc. erkennen muss. Somit ist ein weiterer wichtiger Punkt, dass diese Trainingsdaten ebenfalls handgeschriebene Bilder enthalten, um zuverlässige Vorhersagen treffen zu können. Hier wurden besonders im Bereich von geometrischen Formen keine richtig verlässlichen Daten gefunden. Es musste daher mit "fast" perfekten, nicht handgezeichneten, Formen gearbeitet werden. Das führt zu dem Problem, dass 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, welche frei zugänglich sind. 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 unbrauchbar.

Daten Vorbereitung

Auch die Vorbereitung der Daten war eine Herausforderung. Für das Lernen eines neuronalen Netzes ist z.B. auch die Farbe der Bilddaten entscheidend. Auch invertierte schwarz-weiß Werte können zu Problemen führen. Da die erstellte Anwendung einen schwarzen Hintergrund und weiße Schrift nutzt, müssen die Trainingsdaten ebenfalls diese Eigenschaften aufweisen, um gute Ergebnisse erzielen zu können. Allerdings sind die Daten dahingehend oftmals unterschiedlich und nicht genormt. Eine Herausforderung war hierbei die Daten so zu bearbeiten, dass sie möglichst gut für den vorliegenden Anwendungszweck geeignet sind.

Die Daten liegen meist in unterschiedlichen Formaten und Formen vor und mussten entsprechend unterschiedlich importiert bzw. gelesen und verarbeitet werden.

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. Es konnte auch nicht nur ein Modell für alle Netze genutzt werden, da jedes Netz individuelle Anpassungen benötigt. Als Grundlage wurde ein Basismodell entwickelt, um dieses dann weiter auf die verschiedenen Daten anzupassen. Die Unterschiede liegen hier hauptsächlich in der Anzahl der Layer und den verfügbaren Hyperparametern.

Ein weiterer Punkt, welcher unterschätzt wurde, ist die benötigte Zeit und Rechenleistung, die das Trainieren einiger Modelle in Anspruch genommen hat. Da Hyperparameter hauptsächlich durch “Trial-and-Error“ angepasst werden mussten und hierfür viele Trainingstests durchlaufen wurden. Je nach Komplexität des Netzes und der anpassbaren Parameter sowie Anzahl an Trainingsdaten dauerte ein Trainingsdurchlauf zwischen einer und fünf Stunden. Auch die Rechenleistung war beschränkt, da die eigenen verfügbaren Ressourcen verwendet wurden. Die Komplexität der Modelle musste deshalb an einigen Punkten eingeschränkt werden. Letztendlich führen diese Faktoren dazu, dass in manchen Szenarien die Netze nicht immer die korrekte Vorhersage treffen. Besonders im Bereich der geometrischen Formen ist das Netz hinsichtlich der Genauigkeit noch nicht perfekt und ungenau, was letztendlich aber auch mit den nicht optimalen Trainingsdaten geschuldet ist.

Ausblick:

In diesem Abschnitt werden zukünftige Verbesserungen, Features/Funktionen und mögliche Technologieintegrationen vorgestellt. Nachdem Funktionen definiert wurden, haben sich während der Umsetzung oder beim Testen der Anwendung weitere Ideen gebildet die später umgesetzt und integriert werden können. Diese Inhalte und Funktionen werden nachfolgend aufgelistet.

Weitere Lerninhalte:

Die Anwendung wurde so aufgebaut das sich mit geringem Aufwand 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 die 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.

1. Musterlösung

Aktuell erhält der Nutzer eine Rückmeldung von dem System, ob die abgegeben Antwort richtig war oder nicht. Als “Quality of Life“ Funktion könnte nach der Rückmeldung des Systems eine Art Musterbild angezeigt werden. Dieses Musterbild repräsentiert dann die optimale Lösung. Hierdurch kann der Nutzer besser nachvollziehen was er falsch gezeichnet hat.

2. Detaillierte Konfiguration von Lernplänen

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

3. Datensatz Erstellung von vertrauenswürdigen Personen

Wie im Kapitel 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.

4. Manuelle Überprüfung

Ein neuronales Netz ist nicht fehlerfrei und es kann durchaus vorkommen, dass falsche Entscheidungen getroffen werden. Hier könnte eine zusätzliche Einstellung eingefügt werden, die es Nutzern ermöglicht nach der Überprüfung seine eigene Antwort noch einmal zu werten. Wird die Antwort entgegen der Entscheidung des Netzes als richtig eingestuft, kann dieses Bild dazu genutzt werden das Modell weiter zu trainieren und zu verbessern, um solche Fehler zukünftig 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 dahingehend, wenn es keine oder nur wenige Datensätze für Machine-Learning gibt.