

**Digit Recognition with Machine Learning**

**Website zur Erkennung von**

**handgeschriebenen Ziffern**

Abgabedatum: Heilbronn, den 28.11.2025

**Von:**

Aimee Elaine Grundei

Matrikelnummer: **3095671**

und

Hannah Kraft

Matrikelnummer: **5237485**

**Inhaltsverzeichnis**

### 1 EINLEITUNG ............................................................................................................. 1

**1.1** **PROJEKTDEFINITION................................................................................................... 1**

**1.2** **PROJEKTBESCHREIBUNG............................................................................................. 1**

### 2 EINFÜHRUNG CNN................................................................................................... 2

**2.1** **ALLGEMEINES................................................................................................................... 2**

**2.2** **ERKLÄRUNG............................................................................................................. 3**

### 3 FRONTEND .............................................................................................................. 5

**3.1** **AUFBAU………............................................................................................................ 5**

**3.2** **HTML UND CSS-CODE.................................................................................................. 6**

**3.3** **JAVASCRIPT............................................................................................................... 9**

### 4 BACKEND ................................................................................................................. 4

**4.1** **FLASK …………………............................................................................................. 11**

**4.2** **DATENBANK ………….............................................................................................. 14**

**4.3** **MODELL.................................................................................................................. 16**

### 5 ANLEITUNG .............................................................................................................. 6

**5.1**  **GIT.......................................................................................................................... 6**

**5.2 PROJEKT AUFSETZEN................................................................................................. 6**

**5.3** **DATABASE................................................................................................................ 6**

### 6 FAZIT ........................................................................................................................ 8

**6.1** **HERAUSFORDERUNGEN ............................................................................................. 8**

**6.2** **AUSBLICK................................................................................................................ 8**

**ANHANG ......................................................................................................................... I**

**A.1** **STARTSEITE ............................................................................................................... I**

**A.2** **STATISTICS-SEITE........................................................................................................ I**

**A.3** **HISTORY-SEITE ...........................................................................................................II**

**A.4** **PROJEKTAUFBAU........................................................................................................II**

**A.5** **PROJEKTAUFBAU........................................................................................................II**

**A.6** **PROJEKTAUFBAU........................................................................................................II**

**A.7** **PROJEKTAUFBAU........................................................................................................II**

**A.8** **PROJEKTAUFBAU........................................................................................................II**

**A.9** **TERTIMAL SCREENSHOTS.............................................................................................IV**

# 1 Einleitung (Aimee Elaine Grundei, Hannah Kraft)

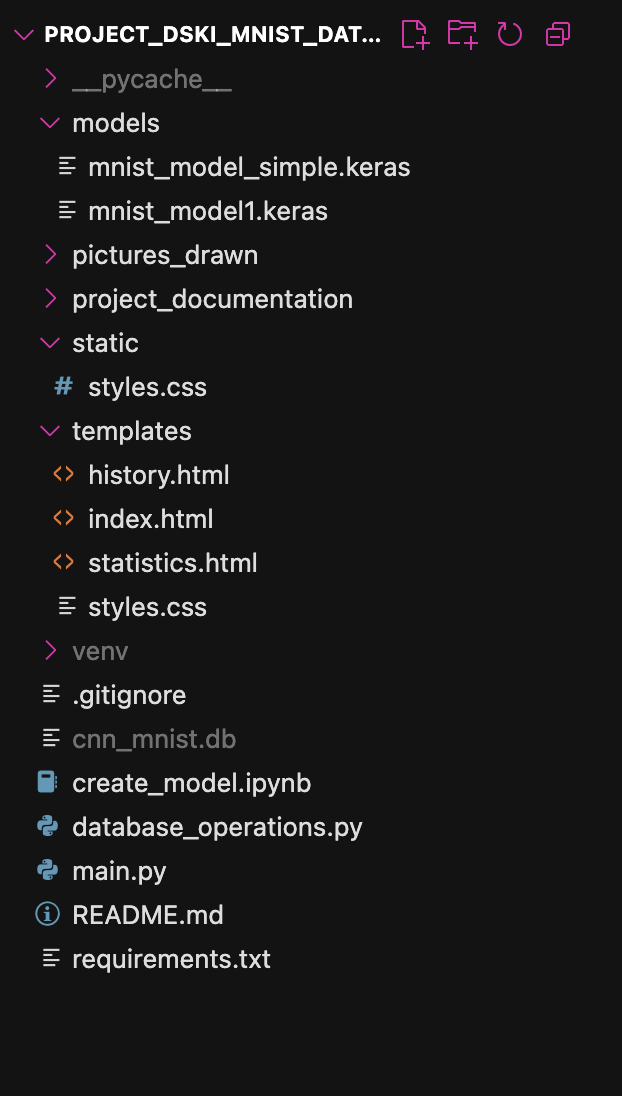
Die folgende Projektdokumentation wurde im Rahmen eines Projekts des Moduls „Grundlagen Datascience und KI“ für das duale Studium „Datascience und Künstliche Intelligenz Smart Operations Management“ erstellt.

### 1.1 Projektdefinition

Im Rahmen des Projekts zur Erkennung von handgeschriebenen Zahlen durch Machine Learning soll eine Webanwendung entwickelt werden. Diese Anwendung ermöglicht das Schreiben von Zahlen, das Ausgeben und Bewerten des Ergebnisses eines Modells, sowie das Darstellen der erfassten Daten als Zahlen, Graphen und Tabelle.

### 1.2 Projektbeschreibung

Die Anwendung wurde anfangs in zwei Teile aufgeteilt, das Frontend und das Backend. Das Frontend beinhaltet den HTML-Code der Seite, sowie das Styling mit Hilfe von CSS. Außerdem wurden einige elementare Funktionen durch Java-Script Code erstellt. Dieser Teil wurde von Hannah Kraft geplant und umgesetzt. Das Backend besteht aus einer Datenbank, die mit SQLite erstellt wurde, der Verarbeitung von Routen durch eine Flask-Anwendung und der Erstellung und dem Training der Machine Learning Modelle. Dieser Teil des Projekts wurde von Aimee Elaine Grundei geplant und umgesetzt.

Um effizient und parallel arbeiten zu können, wurde Git als Versionsverwaltung genutzt. Hierfür wurde ein Git-Repository erstellt, auf das beide Projekterstellerinnen Zugriff haben. Somit konnten beide auf ihrem eigenen Branch arbeiten und die Ergebnisse später auf dem Main-Branch zusammenfassen. Durch die klare Einteilung in Front- und Backend, konnten die einzelnen Aufgaben und Dateien abgegrenzt werden und es gab es keine Merge-Konflikte beim Zusammenführen. Es mussten hier allein die Form der Übergabewerte zwischen HTML und den Flask-Funktionen besprochen werden, sodass eine nahtlose Anwendung erstehen kann.

Die Struktur des Projekts wurde wie das einer klassischen Webanwendung gewählt. Zum Setup wurde eine virtuelle Umgebung und eine „.gitignore“, eine requirements.txt für die notwendigen Installationen und eine README.md zur Beschreibung und Erklärung der Funktionsweise erstellt. Der eigentliche Code besteht aus einer „main.py“, über die das Projekt läuft, den Ordnern „templates“ und „static“ für die HTML-Templates und das Styling. Für die Logik der Anwendung werden außerdem eine Datenbankdatei, eine Datei für die Datenbankoperationen, sowie ein Notebook zur Erstellung der Modelle und ein daraus entstehender Ordner „models“, in dem diese gespeichert werden. Zuletzt werden die auf der Seite gezeichneten Bilder in „pictures\_drawn“ gespeichert, um später mit diesen erneut die Modelle trainieren zu können.

# 2 Einführung CNN

### 2.1 Allgemeines

### 2.2 Erklärung

**3 Frontend**

### 3.1 Aufbau

### 3.2 HTML- und CSS-Code

#### **3.2.1 „HTML**

#### **3.2.2 CSS**

### 3.3 Java-Script

# 4 Backend (Aimee Elaine Grundei)

Das Backend besteht aus einer Datei „main.py“, in der mit Flask die Verbindung zum Frontend umgesetzt ist, einer Datei „database\_operations.py“ mit den Datenbankoperationen und einem Notebook „create\_model.ipynb“, welches die Modelle erstmals erstellt und trainiert.

Im Folgenden soll näher auf die einzelnen Dateien eingegangen und einzelne Codeausschnitter erklärt werden.

### 4.1 Flask

Für die Schnittstelle zwischen dem Frontend und der Logik für die Erkennung der Zahl wurde die Python Library „Flask“ genutzt. Die Nutzung von Flask bietet mehrere Vorteile. Unter anderem kann so eine einfache Kommunikation zwischen der Datenbank, dem Modell und den HTML-Templates gewährleistet werden.

In der Datei „main.py“ befinden sich 5 Funktionen, die jeweils eine Route der Seite abfangen. Beim Starten der App wird einmal eine Flask-App erstellt und ein „secret\_key“ gesetzt, der für gesicherte Sessions sorgt und beispielsweise Login-Status oder flash-Nachrichten in den Cookies sicher verschlüsselt. Als nächstes wird die Tabelle, falls noch nicht vorhanden, in der Datenbank erstellt. Außerdem wird hier bereits für das beim Start eingetragene Modell der richtige Pfad ausgewählt und das Modell geladen.

Im Anschluss wird automatisch die „index“-Route ausgeführt, die das zugehörige Template aufruft. Hierfür wird die Flask-Funktion „render\_template()“ genutzt, da diese automatisch nach der zugehörigen Datei im Ordner „templates“ sucht und Platzhalter, die hier aber noch nicht verwendet werden, durch Werte ersetzt, die aus der Funktion als Schlüsselwort-Argumente übergeben wurden. Am Ende gibt sie so das fertige HTML als String zurück.

A screenshot of a computer program

AI-generated content may be incorrect.Die Seite „history“ soll eine gefüllte Tabelle anzeigen und benötigt deshalb beim Aufruf Daten aus der Datenbank. Dafür werden in der dazugehörigen Funktion zuerst alle Einträge aus der Datenbank geholt und danach so vorbereitet, dass das Template sie benutzen kann. (*Figure 1 Funktion history()*)

Es wird über jeden Eintrag (ein Eintrag entspricht einer Reihe) der Datenbank iteriert, die benötigten Werte in einem dictionary gespeichert und zum Schluss einer Liste angefügt. Somit befindet sich in dieser Liste, die an das Template übergeben wird, für jeden Eintrag ein dictionary mit folgenden Werten: model\_name, predicted, actual, correct, confidence.

Auf der Seite „statistics“ sollen einmal Zahlenwerte zu den insgesamten getroffenen Vorhersagen, den richtigen Vorhersagen und der Gesamtgenauigkeit des Modells angegeben werden. Außerdem sollen zwei Balkendiagramme angezeigt werden, eins zu den totalen Vorhersagen pro Zahl und ein weiteres zu der Genauigkeit pro Zahl. (*A.2 Statistics-Seite*)

Da es mehrere verschiedene Modelle gibt, kann der User hier zwischen mehreren auswählen oder sich die Daten für alle zusammen anzeigen lassen. Diese Wahl wird in der Funktion zuerst angefragt, falls nichts ausgewählt wurde, wird die Berechnung mit allen durchgeführt.

Als nächstes werden alle Daten aus der Datenbank geholt und eine Liste aller Modelle erstellt, die für das Auswahlmenü auf der Seite benötigt wird. Die folgenden Schritte sind aufgeteilt und werden entweder für alle Modelle oder nur für eins ausgeführt. (*Figure 2 Funktion statistics(), get selected model and database entrys*)

Um die Anzahl der gesamten und richtigen Ergebnisse zu zählen, wird jede einzelne Zeile aus dem Datenbankergebnis betrachtet und mit Hilfe eines Dictionarys gezählt, wie oft jede Zahl jeweils vorhergesagt wurde, wie oft eine Zahl gemalt wurde und wie viele richtige Vorhersagen getroffen wurden. Wenn alle Modelle betrachtet werden, werden diese Statistiken pro Modell gespeichert, wodurch danach für jedes Modell einzeln die als nächstes beschriebenen Berechnungen erfolgen.

A computer screen with text on it

AI-generated content may be incorrect.Um die Genauigkeit in Prozent zu bestimmen, wird über die Ziffern null bis neun iteriert, für jede Ziffer die Anzahl der korrekten Vorhersagen durch die tatsächliche Häufigkeit dieser Zahl geteilt und mit hundert multipliziert. Die gesamte Genauigkeit über alle Zahlen wird ebenfalls auf diese Weise berechnet, allerdings hier nicht für jede Zahl einzeln, sondern für alle zusammen.

Zum Schluss wird das Template dann mit den benötigten Werten gefüllt und gerendert.

Die Hauptfunktion des Projektes ist das Erkenne der handgeschriebenen Zahlen. Hierfür wird die Route „predict“ aufgerufen, die selbst kein Template hat, sondern lediglich beim Betätigen des „Submit“-Buttons der Startseite ausgeführt wird, am Ende der Berechnungen aber lediglich ein json-Objekt zurückgibt. Dieses soll am Ende die vom Modell vorhergesagte Zahl beinhalten und angeben, zu wie viel Prozent es sich bei der Bestimmung sicher ist.

Hierfür wird zuerst das gemalte Bild von der Seite erfragt und anschließend aufbereitet. Diese Aufbereitung beinhaltet folgende Schritte: Das gemalte in Grautöne umwandeln, falls nötig die Farben invertieren (das Canvas hat meist schwarze Zahlen auf weißem Hintergrund, unsere Modelle benötigen aber weiße Zahlen auf schwarzem Hintergrund) und Rauschen entfernen. (*Figure 6 Funktion predict(), get image and first preprocessing*)

Außerdem wird das Bild zugeschnitten und zentriert, indem zuerst alle nicht leeren Pixel gesammelt werden, das Minimum und Maximum auf beiden Achsen bestimmt und anhand dieser entstehenden Bounding-Box das Bild geschnitten wird. (*Figure 7 Funktion predict(), cut picture to bounding box*)

Zusätzlich werden die Seiten im Anschluss so erweitert, dass das Bild wieder quadratisch ist und eine Größe von 28x28 hat, damit das Modell es später verarbeiten kann. Zuletzt wird das Bild in ein Numpy-Array umgewandelt, normalisiert und in die richtige Form für das gerade ausgewählte Modell gebracht. (*Figure 8 Funktion predict(), normalize and reshape image()*)

A computer screen with text

AI-generated content may be incorrect.Nach der Vorverarbeitung kann nun die gemalte Zahl in das Modell gegeben werden, wodurch am Ende für jede Ziffer eine Wahrscheinlichkeit zurückgegebenen wird. Hier wird die Zahl mit der höchsten Prozentzahl genommen und im json-Format als Prediction zurück an das Frontend gegeben.

Die letzte Funktion der Seite, ist die des Feedbacks. Nachdem der User die Vorhersage seiner gemalten Zahl bekommt, hat er die Möglichkeit ein Feedback abzugeben. Hierfür gibt er an, welche Zahl er in Wirklichkeit gemalt hat und drückt auf den Button „Submit Feedback“. Dadurch wird im Backend die Funktion „feedback()“ aufgerufen.

A computer screen with text

AI-generated content may be incorrect.Diese Funktion fragt zuerst alle benötigten Daten (das gemalte Bild, das vorhergesagte Lable, das echte Lable, die Confidence) ab und vergleicht anschließend, ob die Vorhersage korrekt war. (*Figure 10 Funktion feedback(), get data from template*) Im Anschluss werden die Bilddaten von einer Liste in ein Image-Objekt umgewandelt und gespeichert. Der Speicherort ist hierbei der Unterordner in „pictures\_drawn“ mit der zugehörigen Zahl. Der Name der Datei hat folgendes Format: „<*Zeitstempel*>\_<*Modelltyp*>\_pred<*Vorhergesagtes* *Lable*>\_ conf<*Confidence*>.png“. (*Figure 11 Funktion feedback(), convert picture to Image and save in files*) Das neue Dataset wird außerdem mit allen wichtigen Werten in die Datenbank hinzugefügt. Als Feedback für die erfolgreiche Speicherung wird beim anschließenden Rendern des Templates der Startseite eine Nachricht mitgesendet. (*Figure 12 Funktion feedback(), save dataset in database and redirect to starting page*)

### 4.2 Datenbank

Um die Statistiken und die Historie anzeigen zu können, müssen die gemalten Zahlen und dazugehörigen Daten gespeichert werden. Dafür wurde in diesem Projekt SQLite genutzt, da es sich um eine kleine und lokale Anwendung handelt. SQLite ist einfach zu integrieren, benötigt keine zusätzliche Konfiguration und eignet sich gut für den Einsatz in Kombination mit Flask. Außerdem ermöglicht es eine schnelle Speicherung und Abfrage der Daten ohne den Aufwand einer komplexen Dantenbanklösung. Durch diese Vorteile ist SQLite sinnvoll, um das Projekt effizient und unkompliziert umzusetzen.

Es werden im Wesentlichen vier Funktionen für die Datenbankoperationen benötigt: das Erstellen, die Speicherung eines neuen Datensatzes, das Holen aller gespeicherter Daten aus der Datenbank und das Filtern nach dem Modelltypen.

A black screen with pink and blue text

AI-generated content may be incorrect.Beim Starten des Programms wird zuerst die Funktion „create\_table“ aufgerufen. (*Figure 13 Funktion create\_table)* Um mit SQLite Datenbankoperationen durchführen zu können, wird immer zuerst eine Verbindung zu dieser benötigt. Diese Verbindung wird durch die Funktion „connect“ erstellt, indem der Pfad zu der Datenbank angegeben wird. Falls die angegebene Datei noch nicht existiert, wird sie mit diesem Befehl zuerst erstellt. Außerdem muss für das Ausführen der Operationen ein Zeiger (cursor) genutzt werden, der über die Verbindung zur Datenbank erstellt wird. Über diesen können jetzt SQL-Statements als Strings geschrieben und durchgeführt werden.

In dieser Funktion wird erstmals eine Tabelle erstellt, falls sie noch nicht existiert, mit den Spalten: id (INTEGER, Primary Key, AUTOINCREMENT), timestamp (TEXT), model\_name (TEXT), input\_data (TEXT), predicted\_label (TEXT), true\_label (TEXT), correct (INTEGER), confidence (REAL).

Die id dient hier der eindeutigen Identifikation eines Sets an Daten und wird automatisch beim Einfügen erstellt. Für diese Anwendung nicht relevante Daten sind der timestamp, der die genaue Zeit der Speicherung angibt, die confidence, die in Prozent angibt, wie sicher sich das Modell bei der Vorhersage war, und input\_data, in der jedes Bild als Array gespeichert ist. Diese Werte dienen im Moment allein der Nachvollziehbarkeit der gespeicherten Daten, könnten für zukünftige Erweiterungen des Programms allerdings wichtig werden.

Der Modellname (model\_name) gibt an, welches Modell für diese Vorhersage genutzt wurde, und ist für die Statistiken wichtig. Für den gleichen Zweck werden außerdem das vorhergesagt Label der gemalten Zahl (predicted\_label) und das echte Label (true\_label) benötigt. Die Spalte correct lässt sich durch Vergleichen aus diesen beiden Labels ableiten, dient also nur der Vereinfachung bei späteren Berechnungen.

Nachdem die Tabelle erstellt wurde, müssen die Änderung gespeichert und die Verbindung geschlossen werden, um andere Verbindungen nicht zu blockieren und RAM-Speicher zu sparen.

Eine weitere zentrale Funktion für das Programm ist das Speichern neuer Daten in der Datenbank bei erhaltenem Feedback des Benutzers. (*Figure 14 Funktion insert\_result()*) Dafür wird ebenfalls zuerst eine Verbindung und ein Zeiger erstellt, der das Einfügen der neuen Zeile ausführt. Hier müssen einmal die Spalten angegeben werden, in die Werte einzufügen sind, und dann die dazugehörigen Werte in gleicher Reihenfolge. Für bessere Lesbarkeit und Wartbarkeit werden hier Fragezeichen als Platzhalter genutzt, die SQLite intern durch die dahinter übergebenen Werte ersetzt. Die Werte, außer die Zeit, die über das Python-Modul „datetime“ geholt wird, werden beim Aufruf der Funktion als Parameter übergeben. Somit muss nicht für jede Insert-Operation eine neue Abfrage erstellt, sondern diese lediglich mit unterschiedlichen Werten gefüllt werden. Zum Schluss wird auch hier die ausgeführte Änderung gespeichert und die Verbindung geschlossen.

Die letzten beiden Funktionen sind für das Holen aller Daten aus der Datenbank für die Statistiken und die Historie, einmal für alle Modelle (*Figure 15 Funktion fetch\_all(), get all data from database*) oder nur für ein einzelnes (*Figure 16 Funktion fetch\_by\_model(), get data for model from database*). Auch hier wird die Verbindung und ein Zeiger bei beiden Varianten erstellt. Danach wird ein SQL-Befehl ausgeführt, der alle Daten aus der angegebenen Tabelle auswählt. Falls nur ein Modell gewollt ist, wird die Tabelle durch Spalte model\_name nach dem angegebenen Modellnamen, der als Parameter der Funktion zu übergeben ist, gefiltert. Im Anschluss werden die eben gewählten Daten aus der Datenbank geholt und die Verbindung wieder geschlossen. Der Rückgabewert der Funktion ist dann die Liste der Werte aus der Datenbank.

### 4.3 Modell

Für das Trainieren des Modells wurde in einem Jupyter Notebook mit dem Mnist-Datensatz der Keras-Bibliothek gearbeitet. Er besteht aus 70.000 Graustufenbildern von handgeschriebenen Ziffern zwischen null und neun und den dazugehörigen Labels. Die Verteilung zwischen den einzelnen Ziffern ist ungefähr gleich, es gibt nur leichte Schwankungen (+/- 10%), daher müssen auch hier die Keras-Daten nicht angepasst werden. Die Bilder sind in der Größe von 28x28 Pixel, wobei ein Pixel eine Zahl von 0 bis 255 enthält und die Helligkeit an dieser Stelle angibt. Dabei steht 0 für schwarz, 255 für weiß und die Werte dazwischen für verschiedene Graustufen. Durch die eben genannten Voraussetzungen eignet sich dieser Datensatz besonders, um möglichst einfach ein Modell zu trainieren, das danach wegen guter Trainingsdaten auch gute Ergebnisse liefert. Es ist nicht viel Vorverarbeitung nötig und es kann sich auf die Richtigkeit der Daten, beispielsweise der Labels, verlassen werden.

Das Notebook startet mit den nötigen Imports der Bibliotheken und dem Setzen der Parameter, die über die gesamte Datei benötigt werden, wie der Anzahl an Klassen, die in diesem Fall aufgrund der zehn verschiedenen Ziffern auch zehn beträgt. Als nächstes werden die Daten aus der Keras Bibliothek geladen und in den Variablen x\_train, x\_test, y\_train und y\_test gespeichert. Die x-Variablen beinhalten hierbei die Bilddaten als Array und die y-Variablen die Labels ebenfalls als Array. Die Aufteilung zwischen den Trainings- und Testdaten ist hier von Keras vordefiniert und beträgt 6:1 (60.000 Trainingsdaten, 10.000 Testdaten). Somit hat das „x\_train“-Array folgende Form: (60000, 28, 28).

Da die Pixeldaten der Bilder ursprünglich zwischen 0 und 255 liegen, werden sie durch die Division durch 255 in den Bereich zwischen 0 und 1 skaliert. Dies erleichtert dem neuronalen Netz das Lernen, da kleinere und einheitlich skalierte Eingabewerte zu einer stabileren und schnelleren Konvergenz während des Trainings führen. Sind die Werte zu groß, können sich die Gewichte und Aktivierungen in den Schichten stark vergrößern und es kann zu „exploding“ oder „vanishing gradients“ kommen, was das Training behindert.

Als nächstes müssen die Zielwerte (Labels) in eine One-Hot-Codierung umgewandelt werden, damit eine implizite Rangordnung der Zahlen vermieden wird und das Modell später die Verlustfunktion „categorical crossentropy“ nutzen kann. Die Label werden also nicht mehr als Zahlen von null bis neun gespeichert, sondern als Vektor der Länge zehn. Jede Position steht hier für eine Zahl, die Position mit der richtigen Klasse hat den Wert eins, alle anderen null. Beispielsweise wird die Ziffer 3 als Label zu: [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

A diagram of a diagram

AI-generated content may be incorrect.Als erstes Modell wurde in einfaches Multi-Layer Perceptron (MLP) gewählt, da es eine einfache Lösung für Bilderkennungsprobleme bietet. Es ist ein klassisches neuronales Netz, das aus mehreren Schichten von Neuronen besteht: einer Eingabeschicht, mindestens einer verdeckten Schicht (Hidden Layer) und einer Ausgabeschicht. Jede Schicht ist vollständig mit der nächsten verbunden und zwischen den Neuronen werden gewichtete Summen gebildet, auf die eine Aktivierungsfunktion angewendet wird. MLPs sind in der Lage, nichtlineare Zusammenhänge zu lernen und somit anders als lineare Modelle komplexe Muster zu erkennen. Allerdings berücksichtigen sie nicht direkt die räumlichen Strukturen eines Bildes, da sie die Daten in einer flachen, eindimensionalen Form entgegennehmen, wobei jede Information über die relative Position von Pixeln zueinander verloren geht. Trotzdem kann es auf kleinen Datensätzen gute Ergebnisse erzielen und wurde deshalb ausgetestet.

A graph of a function

AI-generated content may be incorrect.Dafür wurden die x-Daten „abgeflacht“ (flattened), sodass jedes Bild zu einem eindimensionalen Vektor der Länge 784 (28x28) wird. Danach wurde das Modell definiert, es hat eine Eingabeschicht mit 128 Neuronen, der Aktivierungsfunktion „ReLu“ und erwartet eine Eingabeform von flachen, 784 langen Vektoren.

ReLu (Rectified Linear Unit) gibt den Eingabewert direkt zurück, wenn er positiv ist, und 0, wenn er negativ ist. Sie wird häufig in neuronalen Netzen eingesetzt, weil sie nichtlinear ist und hilft, das Problem verschwindender Gradienten zu reduzieren. Allerdings können durch das Nullsetzen von negativen Zahlen die Gewichtungen einzelner Neuronen ebenfalls Null sein und somit sterben sie ab, tragen also nichts mehr zum Lernprozess bei. Für einzelne Neuronen ist dies kein Problem, allerdings zeigt sich, dass durch ReLu bis zu 50 % der Neuronen wegfallen können, was das Lernen sehr schwierig macht.

A graph of a function

AI-generated content may be incorrect.Die zweite Schicht nutzt ebenfalls ReLu, hat allerdings nur 64 Neuronen, um das Modell kompakt zu halten und so Overfitting zu vermeiden. Außerdem extrahiert bereits die erste Schicht viele grundlegende Merkmale, worauf die zweite Schicht aufbauen und komplexere Zusammenhänge lernen kann.

Die Ausgabeschicht hat so viele Neuronen, wie es Klassen in den Daten gibt. Es werden hier die Ausgaben in Wahrscheinlichkeiten für jede Klasse (0-9) umgewandelt und somit die endgültige Vorhersage getätigt. Dafür wird die Aktivierungsfunktion Softmax verwendet. Sie berechnet für jede Klasse einen Wert zwischen 0 und 1, sodass die Summe aller Ausgaben 1 ergibt. Die Klasse mir der höchsten Wahrscheinlichkeit wird dann als Vorhersage gewählt.

Nach der Erstellung des Modells wird es zuerst für das Training konfiguriert. Dabei wird der Optimierungsalgorithmus (Optimizer) Adam verwendet. Dieser ist für das Anpassen der Gewichte während des Trainings verantwortlich und kombiniert Vorteile von Momentum und adaptiver Lernrate, wodurch das Training schneller und stabiler konvergiert.

A graph showing a function

AI-generated content may be incorrect.Adam startet mit zufällig gewählten Gewichten in das Modell und soll die Parameter finden, die die Zielfunktion, die misst, wie gut die Vorhersage eines Modells ist, minimieren. Dafür werden die Gewichte nach und nach aktualisiert, immer in die Richtung des stärksten Gefälles (Gradientenabstiegsverfahren). Die Größe der Änderung, die Lernrate, bestimmt Adam dynamisch, je nachdem wie stark das Gefälle ist, je größer, desto kleinere Schritte werden verwendet. Dies ist das bereits angesprochene adaptive Lernen und ein großer Vorteil von Adam. Er merkt sich außerdem die letzten Schritte und passt die Geschwindigkeit des Lernens an, wenn länger in die gleiche Richtung geändert wird. Hier spricht man von einem Momentum, das dabei hilft, den Lernprozess zu beschleunigen. Zudem merkt sich Adam die Informationen über die bereits durchlaufene Umgebung und passt sie im Laufe des Trainings an, sodass effizientere Schritte gemacht werden können.

Die zu minimierende Funktion ist die Loss-Funktion, die angibt, wie gut das Modell die Vorhersagen trifft. Da eine mehrklassige Klassifikation mit One-Hot-codierten Lables vorliegt, wird die kategoriale Kreuzentropie (categorical crossentropy) verwendet. Sie misst die Differenz zwischen den Vorhergesagten Wahrscheinlichkeiten und den tatsächlichen Labels. Zusätzlich zu dem Loss wird während des Trainings und der Auswertung die Genauigkeit berechnet, also der Anteil der korrekt klassifizierten Beispiele, um den Trainingsstand zu überwachen.

Zum Schluss wird dann das Modell trainiert. Dafür müssen die x- und y-Trainingdaten in das Modell gegeben werden. Außerdem kann hier die Anzahl der Durchläufe durch alle Daten, die Epochen, die Größe der Mini-Batches und der Validierungssplit gewählt werden. Die Batch-Größe gibt dabei an, in wie viele kleine Untergruppen (Batches) die Daten aufgeteilt werden. Nach jedem Batch werden dann die Gewichte aktualisiert. Der Validierungssplit gibt die Prozentzahl der Trainingsdaten an, die nicht ins Training einfließen, sondern genutzt werden, um die Leistung des Modells während des Trainings zu überwachen und beispielsweise Overfitting zu erkennen.

Das Feedback, das nach jeder Epoche erstellt wird, sieht wie folgt aus:

A black screen with white text

AI-generated content may be incorrect.

Hier sieht man eine gute Lernkurve, bei der die accuracy und validation accuracy nicht weit auseinander liegen. Um die Güte des Modells richtig bewerten zu können, wird die Genauigkeit mit den Testdaten evaluiert. Hier liegt diese bei 97,58 Prozent, also sogar höher als bei den Trainingsdaten.

Im Anschluss wird das Modell gespeichert und kann in der Anwendung zur Erkennung der manuell geschriebenen Zahlen eingesetzt werden.

# 

# 5 Anleitung (Aimee Elaine Grundei, Hannah Kraft)

In diesem Abschnitt soll erklärt werden, wie das Projekt aufgesetzt wurde und wie es von Dritten genutzt werden kann. Zusammengefasst muss dazu zuerst das Git-Repository geklont, danach eine virtuelle Umgebung erstellt und die nötigen Installationen durchgeführt werden. Als letztes kann dann das Projekt gestartet werden.

### 5.1 Git

Das Git-Repository ist öffentlich und kann deshalb von jedem angesehen und auch geklont werden. Es ist unter folgendem Link zu finden:

[*https://github.com/AimeeElaineGrundei/project\_dski\_mnist\_dataset.git*](https://github.com/AimeeElaineGrundei/project_dski_mnist_dataset.git)*.*

Um das Projekt lokal ausführen zu können, muss es in einen Ordner auf dem Zielmedium geklont werden. Hierzu muss folgende Zeile im Terminal ausgeführt werden:

*git clone* [*https://github.com/AimeeElaineGrundei/project\_dski\_mnist\_dataset.git*](https://github.com/AimeeElaineGrundei/project_dski_mnist_dataset.git)

Nach der Zeile sollte sich das vollständige Projekt in dem ausgewählten Ordner befinden. Um den aktuellen Stand der Anwendung zu bekommen, kann jetzt in Zukunft im Terminal ab dem Ordner „project\_dski\_mnist\_dataset“ folgende Zeile ausgeführt werden:

*git pull*

### 5.2 Projekt aufsetzen

Nach dem Klonen des Projektes, soll als nächstes eine virtuelle Umgebung erstellt und aktiviert werden. Dafür ist folgender Code im Terminal ab dem Verzeichnis „project\_dski\_mnist\_dataset“ auszuführen (empfohlene Pythonversion: 3.13):

*python3.13 -m venv <name\_der\_venv>*

*source <name\_der\_venv>/bin/activate*

In der aktivierten „venv“ sollen jetzt die nötigen Pakete installiert werden, die in der „requirements.txt“ zusammengefasst sind. Mit Hilfe von pip kann dies wie folgt für alle Pakete zusammen geschehen):

*pip install -r requirements.txt*

Mit erfolgreicher Installation wurden alle nötigen Vorbereitungen getroffen und die Anwendung kann gestartet werden:

*python3 main.py*

### 5.3 Database

Die Datenbank wird nicht von Git verwaltet, wird allerdings automatisch beim Start der App erstellt, falls in dem Ordner noch nicht vorhanden. Zum Öffnen, Betrachten und Bearbeiten der Datenbank kann der „DB Browser for SQLite“ verwendet werden. Dieser kann mit folgendem Befehl heruntergeladen werden:

*brew install db-browser-for-sqlite*

# 6 Fazit (Aimee Elaine Grundei, Hannah Kraft)

Insgesamt hat das Projekt gezeigt, wie vielfältig die Schnittstellen zwischen Machine Learning und Webentwicklung sind. Trotz einiger technischer Herausforderungen konnte ein funktionierendes, didaktisch wertvolles System umgesetzt werden.

### 6.1 Herausforderungen

Im Verlauf des Projekts traten verschiedene technische und organisatorische Herausforderungen auf, die sowohl das maschinelle Lernen als auch die Webentwicklung betrafen.

Da das Projekt zu zweit durchgeführt wurde, lag ein besonderer Fokus auf der klaren Aufgabenteilung und der kontinuierlichen Abstimmung zwischen Frontend und Backend.

Die Arbeit mit Git stellte anfangs eine Herausforderung dar, da Versionskonflikte und unterschiedliche Entwicklungsumgebungen koordiniert werden mussten. Durch regelmäßige Abstimmung konnten diese Schwierigkeiten jedoch schrittweise überwunden werden.

Eine zentrale technische Herausforderung war die Verarbeitung der vom Nutzer gezeichneten Zahlen, sodass diese im richtigen Format an das Modell übergeben werden konnten. Dies erforderte eine präzise Vorverarbeitung der übergebenen Werte aus der HTML-Canvas.

Auch die Erstellung leistungsfähiger Modelle stellte eine Herausforderung dar. Trotz Optimierung der Modelle zeigte sich, dass selbst bei guten Ergebnissen auf dem MNIST-Testdatensatz Abweichungen in der Webanwendung auftraten, insbesondere bei untypisch oder unsauber gezeichneten Zahlen. Dies verdeutlichte den Unterschied zwischen idealen Trainingsdaten und realen Nutzereingaben.

Ein weiterer Schwerpunkt lag auf der Speicherung der Nutzereingaben in einer SQLite-Datenbank. Dabei war es wichtig, sowohl die Vorhersagen als auch die tatsächlichen Benutzerangaben konsistent zu erfassen, um im Anschluss aussagekräftige statistische Auswertungen zu ermöglichen.

### 6.2 Ausblick

Das Projekt bietet verschiedene Möglichkeiten für eine Weiterentwicklung.

Ein naheliegender nächster Schritt wäre die Erweiterung der Datensammlung um weitere, realistische Nutzereingaben. Dadurch könnte das Modell kontinuierlich nachtrainiert und seine Robustheit gegenüber handgezeichneten Zahlen verbessert werden. Langfristig ließe sich so ein aktives Lernsystem realisieren, das aus neuen Benutzerdaten selbstständig dazulernt.

Auf technischer Ebene bestehen ebenfalls Optimierungsmöglichkeiten. So könnte der Einsatz eines moderneren Frameworks wie FastAPI die Performance und Wartbarkeit des Backends erhöhen. Im Frontend wären zudem eine noch intuitivere Benutzerführung und direktes Feedback, beispielsweise durch visuelle Markierung richtiger oder falscher Vorhersagen, denkbar.

Darüber hinaus kann das Projekt in Richtung eines Lehr- und Demonstrationstools weiterentwickelt werden, das Studierenden anschaulich vermittelt, wie neuronale Netze trainiert, integriert und bewertet werden – von der Datenerfassung bis hin zur interaktiven Anwendung.

# Anhang

## A.1 Startseite

A screenshot of a computer game

AI-generated content may be incorrect.

## A.2 Statistics-Seite

## A screenshot of a graph AI-generated content may be incorrect.

A graph of different sizes and colors

AI-generated content may be incorrect.

## A.3 History-Seite

A screenshot of a computer

AI-generated content may be incorrect.

## A.4 Projektaufbau

## A screenshot of a computer program AI-generated content may be incorrect.

## A.5 Funktion „def history()“

## A screenshot of a computer program AI-generated content may be incorrect.

Figure 1 Funktion history()

## A.6 Funktion „def statics()“

A computer screen with text

AI-generated content may be incorrect.

Figure 2 Funktion statistics(), get selected model and database entrys

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 3 Funktion statistics(), get data for all models

A computer code on a black background

AI-generated content may be incorrect.

Figure 4 Funktion statistics(), get data for one model

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure 5 Funktion statistics(), get data for one model

## A.7 Funktion „def predict()“

A computer screen with text

AI-generated content may be incorrect.

Figure 6 Funktion predict(), get image and first preprocessing

A computer screen with text

AI-generated content may be incorrect.

Figure 7 Funktion predict(), cut picture to bounding box

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 8 Funktion predict(), normalize and reshape image()

A computer screen with text on it

AI-generated content may be incorrect.

Figure 9 Funktion predict(), make predictions with model

## A.8 Funktion „def feedback()“

## A computer code with text AI-generated content may be incorrect.

Figure 10 Funktion feedback(), get data from template

A computer screen with text

AI-generated content may be incorrect.

Figure 11 Funktion feedback(), convert picture to Image and save in files

A computer screen shot of a black background

AI-generated content may be incorrect.

Figure 12 Funktion feedback(), save dataset in database and redirect to starting page

## A.9 Datenbankfunktionen

A computer screen with text on it

AI-generated content may be incorrect.

Figure 13 Funktion create\_table()

A computer screen with text on it

AI-generated content may be incorrect.

Figure 14 Funktion insert\_result()

## A screenshot of a computer AI-generated content may be incorrect.

Figure 15 Funktion fetch\_all(), get all data from database

A screenshot of a computer

AI-generated content may be incorrect.

Figure 16 Funktion fetch\_by\_model(), get data for model from database

## A.10 „create\_model“- Notebook

## A screenshot of a computer AI-generated content may be incorrect.

Figure 17 Notebook, imports, parameters and load data

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 18 Notebook, scale data

A black screen with white text

AI-generated content may be incorrect.

Figure 19 Notebook, convert labels

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 20 Notebook, transform x and define model

A screenshot of a computer

AI-generated content may be incorrect.

Figure 21 Notebook, compile and train model

## A.6 HTML-Code xy

## A.7 Java-Script-Code

## A.8 CSS

## A.9 Terminal-Screenshots

A black screen with white text

AI-generated content may be incorrect.

Figure 22 Git-Repository klonen

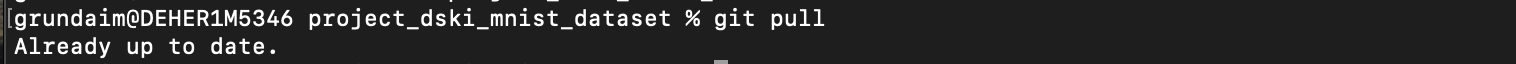


Figure 23 pull-Request

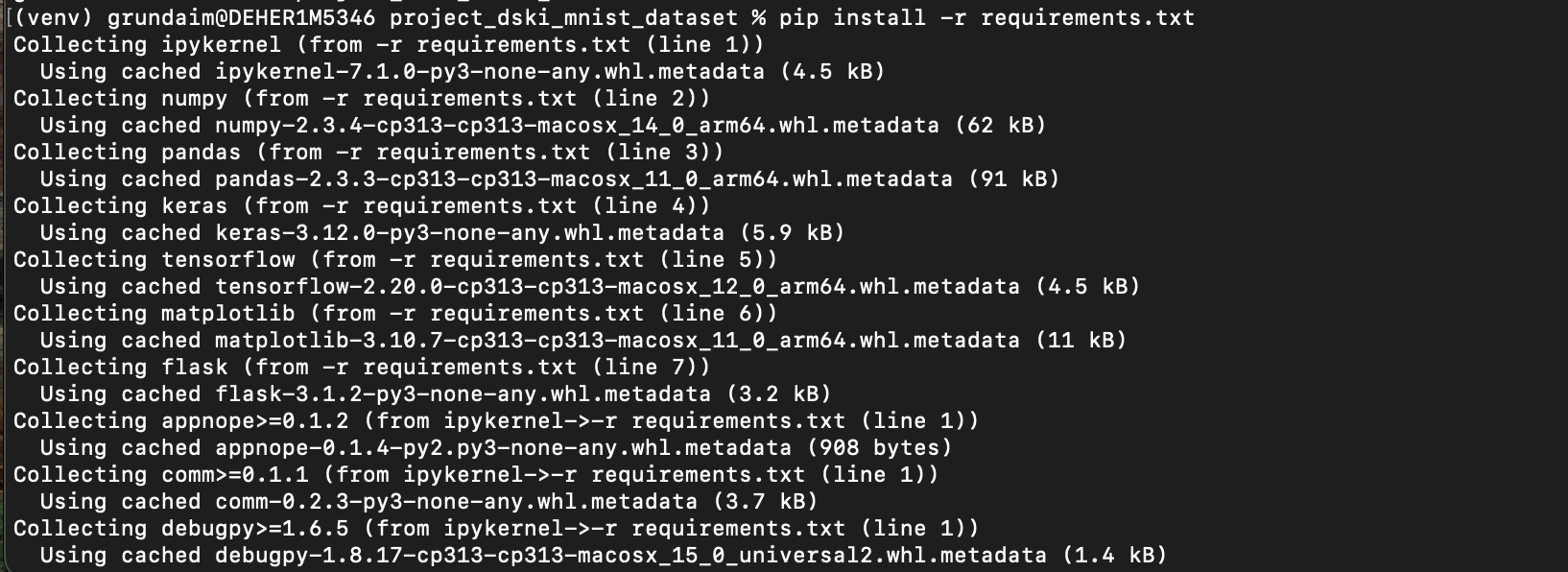


Figure 24 Requirements installieren

A screen shot of a computer

AI-generated content may be incorrect.

Figure 25 App starten

# Quellen

* Foto ReLu: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fde.linkedin.com%2Fpulse%2Factivation-functions-neural-networks-leonardo-calderon-j-%3Ftl%3Dde&psig=AOvVaw1Vlye6Y0vBtwbiRTWEA2my&ust=1763147566223000&source=images&cd=vfe&opi=89978449&ved=0CBUQjRxqFwoTCKCogdrq75ADFQAAAAAdAAAAABAE>
* Foto MLP: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fmichaelkipp.de%2Fdeeplearning%2FFeedforwardNetze.html&psig=AOvVaw2BKygdhs2ygig3qPHSx97P&ust=1763147835066000&source=images&cd=vfe&opi=89978449&ved=0CBUQjRxqFwoTCIDbwtTr75ADFQAAAAAdAAAAABAE>
* Aktivierungsfunktionen – ReLu: <https://databasecamp.de/ki/relu>
* Aktivierungsfunktionen – Softmax: <https://databasecamp.de/ki/softmax>
* Foto Softmax: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fbotpenguin.com%2Fglossary%2Fsoftmax-function&psig=AOvVaw3OYM2asYVJGwzW4vRzHtu8&ust=1763149297436000&source=images&cd=vfe&opi=89978449&ved=0CBUQjRxqFwoTCLDyyJLx75ADFQAAAAAdAAAAABAs>
* Optimierungsfunktion – Adam: <https://databasecamp.de/ki/adam-optimizer>
* Foto Adam: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40anshm18111996%2Fcomprehensive-overview-optimizers-in-machine-learning-and-ai-57a2b0fbcc79&psig=AOvVaw1OsAmsqKnRFIsE0BjAXRA7&ust=1763150402987000&source=images&cd=vfe&opi=89978449&ved=0CBUQjRxqFwoTCLjdlJ7175ADFQAAAAAdAAAAABAT>