

**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 ................................................................................................................. 3

**4.1** **FLASK …………………............................................................................................. 3**

**4.2** **DATENBANK ………….............................................................................................. 6**

**4.3** **MODELL.................................................................................................................. 7**

### 5 ANLEITUNG .............................................................................................................. 13

**5.1**  **GIT......................................................................................................................... 13**

**5.2 PROJEKT AUFSETZEN................................................................................................ 13**

**5.3** **DATABASE............................................................................................................... 13**

### 6 FAZIT ....................................................................................................................... 14

**6.1** **HERAUSFORDERUNGEN ............................................................................................ 14**

**6.2** **AUSBLICK............................................................................................................... 14**

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

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

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

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

**A.4** **PROJEKTAUFBAU.......................................................................................................III**

**A.5** **FUNKTION „DEF HISTORY()“..........................................................................................IV**

**A.6** **FUNKTION „DEF STATISTICS()“......................................................................................IV**

**A.7** **FUNKTION „DEF PREDICT()“..........................................................................................VI**

**A.8** **FUNKTION „DEF FEEDBACK()“......................................................................................VII**

**A.9** **DATENBANKFUNKTIONEN...........................................................................................VIII**

**A.10** **CREATE\_MODEL-NOTEBOOK......................................................................................X**

**A.11** **HTML-CODE...........................................................................................................XV**

**A.12** **JAVA-SCRIPT\_CODE................................................................................................XV**

**A.13** **CSS.....................................................................................................................XVI**

**A.14** **TERTIMAL SCREENSHOTS.......................................................................................XVI**

**A.15** **QUELLEN.............................................................................................................XVII**

# 

# 1 Einleitung (Aimee Elaine Grundei, Hannah Kraft)

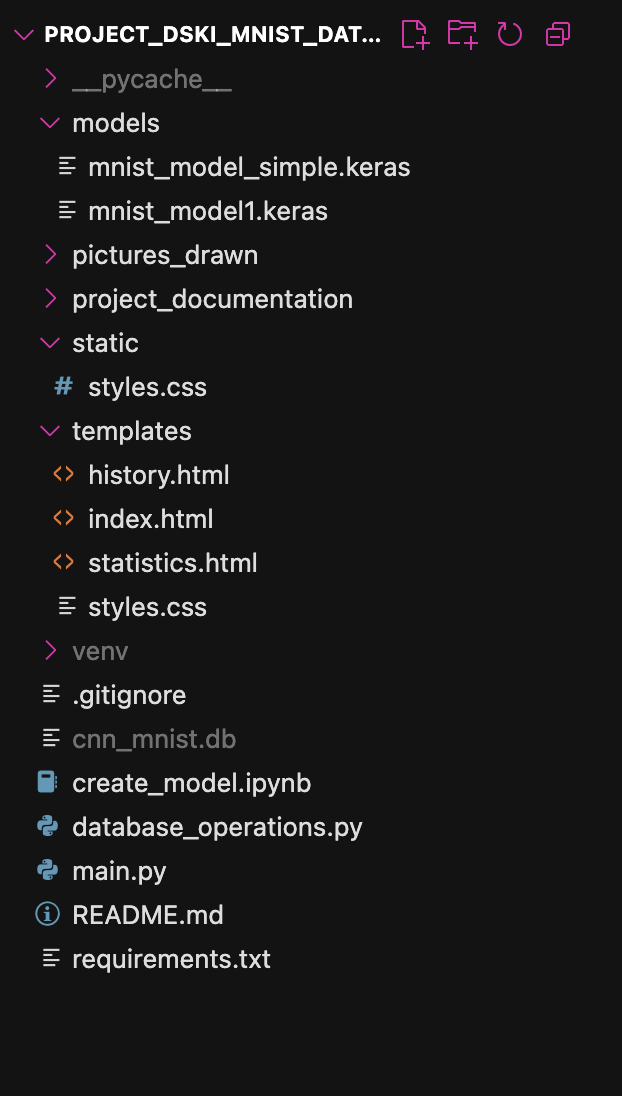
Die folgende Projektdokumentation wurde im Rahmen eines Projekts des Moduls „Grundlagen Datascience und KI“ für das duale Studium „Datascience und Künstliche Intelligenz Smart Operations Management“ erstellt.

### 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 Codeausschnitte 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 fünf 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 Schlüsselwortargumente 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 insgesamt 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 Dictionaries 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 Erkennen 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 (*Figure 6 Funktion predict(), get image and first preprocessing*):

* Das Bild wird in Graustufen umgewandelt
* Falls nötig, werden die Farben invertiert (das Canvas hat meist schwarze Zahlen auf weißem Hintergrund, unsere Modelle benötigen aber weiße Zahlen auf schwarzem Hintergrund)
* Für das optimierte CNN wird zusätzlich das Rauschen entfernt

Außerdem wird nur für das optimierte CNN 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. Für das MLP und das normale CNN dürfen diese Schritte nicht erfolgen, da diese auf unbearbeiteten Bildern trainiert wurden und auch nur genau diese erkennen können. (*Figure 7 Funktion predict(), cut picture to bounding box*)

Zusätzlich wird das Bild anschließend so erweitert, dass es wieder quadratisch ist und eine Größe von 28x28 Pixel 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 aktuell 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 die gemalte Zahl nun in das Modell gegeben werden, wodurch für jede Ziffer eine Wahrscheinlichkeit zurückgegeben wird. Die Zahl mit der höchsten Wahrscheinlichkeit wird dann als Vorhersage genommen und im JSON-Format an das Frontend übermittelt.

Die letzte Funktion der Seite ist die Feedback-Funktion. Nachdem der User die Vorhersage seiner gemalten Zahl bekommt, hat er die Möglichkeit, 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.

Diese Funktion führt folgende Schritte aus:

1. Zuerst werden alle benötigten Daten abgefragt: das gemalte Bild, das vorhergesagte Label, das echte Label und die Confidence. Anschließend wird geprüft, ob die Vorhersage korrekt war. (*Figure 10 Funktion feedback(), get data from template*)
2. Die Bilddaten werden von einer Liste in ein Image-Objekt umgewandelt und gespeichert. Der Speicherort befindet sich im Unterordner „pictures\_drawn“ mit der zugehörigen Zahl. Der Dateiname hat folgendes Format:

„<Zeitstempel>\_<Modelltyp>\_pred<Vorhergesagtes Label>\_conf<Confidence>.png“ (*Figure 11 Funktion feedback(), convert picture to Image and save in files*)

1. Das neue Dataset wird außerdem mit allen wichtigen Werten in die Datenbank hinzugefügt.
2. 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 die 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 Datenbanklösung. Durch diese Vorteile ist SQLite sinnvoll, um das Projekt effizient und unkompliziert umzusetzen.

Es werden im Wesentlichen vier Funktionen für die Datenbankoperationen benötigt:

* Erstellen der Datenbanktabelle
* Speicherung eines neuen Datensatzes
* Holen aller gespeicherten Werte aus der Datenbank
* Filtern nach Modelltyp

Beim Starten des Programms wird zuerst die Funktion „create\_table“ aufgerufen. (*Figure 13 Funktion create\_table)*

A black screen with pink and blue text

AI-generated content may be incorrect.Um mit SQLite Datenbankoperationen durchführen zu können, wird zuerst eine Verbindung zur Datenbank erstellt. Diese Verbindung wird durch die Funktion connect hergestellt, indem der Pfad zur Datenbank angegeben wird. Falls die angegebene Datei noch nicht existiert, wird sie mit diesem Befehl erstellt.

Außerdem wird für das Ausführen der Operationen ein Cursor (Zeiger) genutzt, der über die Verbindung zur Datenbank erstellt wird. Über diesen Cursor können SQL-Statements als Strings geschrieben und ausgeführt 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 Datensatzes 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 der Nachvollziehbarkeit der gespeicherten Daten, könnten aber für zukünftige Erweiterungen wichtig werden.

Der Modellname (model\_name) gibt an, welches Modell für diese Vorhersage genutzt wurde, und ist für die Statistiken wichtig. Für denselben Zweck werden außerdem das vorhergesagt Label (predicted\_label) und das echte Label (true\_label) benötigt. Die Spalte correct lässt sich durch Vergleichen dieser beiden Labels ableiten und dient der Vereinfachung späterer Berechnungen.

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

Eine weitere zentrale Funktion ist das Speichern neuer Daten, wenn der Benutzer Feedback gibt. (*Figure 14 Funktion insert\_result()*)

Dafür wird ebenfalls zuerst eine Verbindung und ein Cursor erstellt, der das Einfügen einer neuen Zeile ausführt. Hierbei müssen Spalten und die dazugehörigen Werte in gleicher Reihenfolge angegeben werden. Für bessere Lesbarkeit und Wartbarkeit werden Platzhalter (?) verwendet, die SQLite intern durch die übergebenen Werte ersetzt. Die Werte, außer der Zeit, die über das Python-Modul datetime ermittelt wird, werden beim Aufruf der Funktion als Parameter übergeben. Dadurch muss nicht für jede Insert-Operation eine neue Abfrage erstellt, sondern diese lediglich mit unterschiedlichen Werten gefüllt werden. Zum Schluss wird die Änderung gespeichert und die Verbindung geschlossen.

Die letzten beiden Funktionen dienen dem Abruf aller Daten aus der Datenbank, entweder 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 Cursor erstellt, anschließend wird ein SQL-Befehl ausgeführt, der alle Daten aus der angegebenen Tabelle auswählt. Falls nur ein Modell abgefragt werden soll, wird die Tabelle über die Spalte model\_name gefiltert. Die Daten werden aus der Datenbank ausgelesen, die Verbindung geschlossen und als Liste von Datensätzen zurückgegeben.

### 4.3 Modell

Für das Trainieren des Modells wurde ein Jupyter Notebook genutzt, das den MNIST-Datensatz der Keras-Bibliothek verwendet. Dieser Datensatz besteht aus 70.000 Graustufenbildern handgeschriebener Ziffern von 0 bis 9 sowie den dazugehörigen Labels. Die Verteilung der einzelnen Ziffern ist ungefähr gleich, es gibt nur leichte Schwankungen (+/- 10%), sodass die Daten nicht weiter angepasst werden müssen. Die Bilder haben eine Größe von 28x28 Pixel, wobei ein Pixel eine Zahl von 0 bis 255 enthält, die die Helligkeit angibt: 0 für schwarz, 255 für weiß und Werte dazwischen für Graustufen. Dank dieser Eigenschaften eignet sich der Datensatz besonders gut, um ein Modell zu trainieren, das robuste Ergebnisse liefert. Eine große Vorverarbeitung ist nicht nötig, und das Modell kann sich auf die korrekten Labels verlassen.

Das Notebook beginnt mit den Imports der benötigten Bibliotheken und dem Setzen wichtiger Parameter, z.B. der Anzahl der Klassen (10). Danach werden die Daten aus Keras geladen und in den Variablen x\_train, x\_test, y\_train, y\_test gespeichert. (*Figure 17 Notebook, imports, parameters and load data*)

* x\_train/x\_test enthalten Bilddaten als Arrays
* y\_train/y\_test enthalten Labels ebenfalls als Array

Die Aufteilung zwischen Trainings- und Testdaten ist hier von Keras vordefiniert und beträgt 60.000 Trainingsbilder/10.000 Testbilder. Das x\_train-Array hat somit die Form (60000, 28, 28).

Die Pixelwerte werden durch Division durch 255 auf den Bereich 0 bis 1 skaliert. (*Figure 18 Notebook, scale data*) Dies erleichtert dem neuronalen Netz das Lernen, verhindert exploding/vanishing gradients und sorgt für eine stabile und schnellere Konvergenz.

Die Labels werden anschließend in One-Hot-Encoding umgewandelt, um eine implizite Rangordnung zu vermeiden. (*Figure 19 Notebook, convert labels*) Jede Ziffer wird so zu einem Vektor der Länge 10, z. B.: 3 → [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 Multi-Layer Perceptron (MLP) gewählt. Es ist ein klassisches neuronales Netz, bestehend aus:

* einer Eingabeschicht
* mindestens einer verdeckten Schicht (Hidden Layer)
* 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 dadurch 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 getestet.

A graph of a function

AI-generated content may be incorrect.Dafür wurden die x-Daten abgeflacht (Flatten), 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. (*Figure 20 Notebook, transform x and define model MLP*)

ReLu (Rectified Linear Unit) setzt negative Werte auf 0 und gibt positive unverändert zurück. Sie wird häufig in neuronalen Netzen eingesetzt, weil sie nichtlinear ist und hilft, das Problem verschwindender Gradienten (vanishing gradients) zu reduzieren. Allerdings können durch das Nullsetzen von negativen Zahlen die Gewichtungen einzelner Neuronen Null werden 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 schwieriger 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. (*Figure 21 Notebook, compile and train model MLP*) 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.

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 A graph showing a function

AI-generated content may be incorrect.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. (*Figure 21 Notebook, compile and train model MLP*) 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.

A screenshot of a computer

AI-generated content may be incorrect.Im Anschluss wird das Modell gespeichert und kann in der Anwendung zur Erkennung der manuell geschriebenen Zahlen eingesetzt werden. (*Figure 22 Notebook, evaluate and save model MLP*) Auf den neuen Daten ist die Leistung des Modells leider nicht mehr besonders gut. Es wurden 100 Zahlen geschrieben, jede Zahl 10 Mal. Das Modell hat hier nur eine Accuracy von 64 Prozent, was daran liegt, dass die gemalten Zahlen nicht immer die ideale Größe oder Position wie im Trainingsdatensatz haben. Da das MLP jeden Pixel als eigenständiges Feature ohne räumlichen Kontext sieht, ist die Position der Zahl aber entscheidend.

Genau dieses Problem löst ein CNN, es nimmt dreidimensionale Arrays an und analysiert auch die relative Position zu den anderen Pixeln. (*Figure 23 Notebook, build model CNN*) Die Eingabegröße für die erste Schicht des Modells ist deshalb (28, 28, 1), 28 für die Pixelhöhe und -breite und 1 für den Farbkanal, also in diesem Fall Graustufen. Die zweite Schicht ist eine „Convolutional Layer“, eine „Faltungsschicht“. Sie nutzt die Aktivierungsfunktion ReLu, 16 Filter der Größe 3x3 und scannt das Bild damit nach simplen Feautures wie Ecken oder Linien. Als nächstes wird in der MaxPooling-Layer die räumliche Größe der Feature Map reduziert, indem das Eingabebild in kleinere Bereiche der Größe 2x2-Pixel aufgeteilt und aus jedem dieser Bereiche der Maximalwert ausgewählt wird. Dadurch entsteht eine kleine, komprimierte Version des Bildes, was den Rechenaufwand reduziert und Robustheit gegen kleine Verschiebungen oder Störungen bietet.

A screenshot of a computer

AI-generated content may be incorrect.Diese beiden Schichten (Conv, MaxPooling) werden noch einmal wiederholt, dabei lernt die „Convolutional Layer“ mit 32 Filtern komplexere Features durch Verbinden der vorherigen Features. Im Anschluss werden in der Flatten-Layer die 2D-Featuremaps in 1D-Feature-Vektoren für die anschließende Dense-Layer konvertiert. Vorher wird allerdings noch eine Drop-Out-Layer verwendet, die während des Trainings 50 Prozent der Neuronen dieser Schicht deaktiviert, um Overfitting zu vermeiden. Die letzte Schicht ist dann die Dense-Layer, die alle Features der Flatten-Layer zu jeder Output-Klasse verbindet und mit der Softmax-Aktivierung die Wahrscheinlichkeitsverteilung über die 10 Klassen ausgibt.

Nach einem 15 Epochen Training hat das Modell auf einen Testdatensatz eine Genauigkeit von 98,82 Prozent, beim Einsetzen in der Webanwendung allerdings nur noch 83 Prozent. (*Figure 24 Notebook, compile and train model CNN*) Das Modell hat immernoch Probleme, Zahlen zu erkennen, die nicht genau die gleiche Form, Größe und Ausrichtung haben, wie im Trainingsdatensatz.

Auf Grund dieser Probleme, soll erneut ein CNN trainiert werden, das optimiert ist und robuster auf die Bilder der realen Anwendung reagiert.

Für das Optimieren wurde Optuna genutzt. Optuna ist ein automatisiertes Hyperparameter-Optimierungstool, das dabei hilft, ein Modell effizienter und besser abzustimmen. Es nutzt intelligente Suchstrategien wie den Tree-structured Parzen Estimator, der anders als Grid Search die Suche wahrscheinlichkeitsbasiert steuert, um schneller vielversprechendere Einstellungen zu finden.

Es wurde eine Funktion objective(trial) erstellt, bei der für die einzelnen Schichten verschiedene Anzahl Neuronen, Aktivierungsfunktionen, Dropout-Rate, Learning-Rate und Optimierungsfunktionen in Variablen definiert wurden, die später getestet werden sollen. (*Figure 26 Notebook, define parameters to test*) Danach wird das Modell erstellt und die Hyperparameter mit den Variablen gefüllt. Abhängig von dem gewählten Optimierer wird dann auch die Lernrate erstellt. Das Modell wird mit categorical\_crossentropy kompiliert und für wenige Epochen trainiert, um die Optimierung effizient zu halten. Während des Trainings wird ein Teil der Daten als Validierungsset genutzt, damit Optuna beurteilen kann, wie gut die gewählten Hyperparameter generalisieren. Am Ende ermittelt die Funktion die beste Validierungsgenauigkeit dieses Trainingsdurchlaufs und gibt sie zurück. Optuna nutzt diesen Wert, um zu entscheiden, welche Hyperparameter in den nächsten Versuchen bevorzugt getestet werden sollen. (*Figure 27 Notebook, define model, choose optimizer and train model*)

Nach der Definition der Funktion wird eine neue Optuna-Studie gestartet, die das Ziel hat, die Hyperparameter so zu wählen, dass die Validierungsgenauigkeit maximiert wird. Mit Ausführen der Studie wird nun die vorherige Funktion aufgerufen und unterschiedliche Hyperparameterkombinationen getestet, bis entweder eine Anzahl von 30 Versuchen oder eine Zeit von einer Stunde erreicht ist. (*Figure 28 Notebook, start optuna study*)

A screenshot of a computer

AI-generated content may be incorrect.Im Anschluss wird das finale Modell mit den besten Parametern der Studie definiert, kompiliert und trainiert. (*Figure 29 Notebook, define and train final model*) Dabei wurde eine Genauigkeit von 99,1 Prozent erreicht. Das Modell konnte gespeichert und in der Anwendung getestet werden. (*Figure 30 Notebook, evaluate and save model CNN optimized*) Auch dort wurde eine Accuracy von 95 Prozent erreicht, was eine deutliche Steigerung zu den anderen beiden Modellen zeigt. Es können jetzt auch Zahlen erkannt werden, die nicht zentriert oder zu groß bzw. zu klein gezeichnet wurden. Das liegt daran, dass die Bilder nun vorverarbeitet werden dürfen, da das Modell robuster gegenüber Veränderungen im Vergleich zum Trainingsdatensatz ist.

# 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 Funktion history()

## A.6 Funktion „def statistics()“

A computer screen with text

AI-generated content may be incorrect.

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

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Funktion statistics(), get data for all models

A computer code on a black background

AI-generated content may be incorrect.

Figure Funktion statistics(), get data for one model

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure Funktion statistics(), get data for one model

## A.7 Funktion „def predict()“

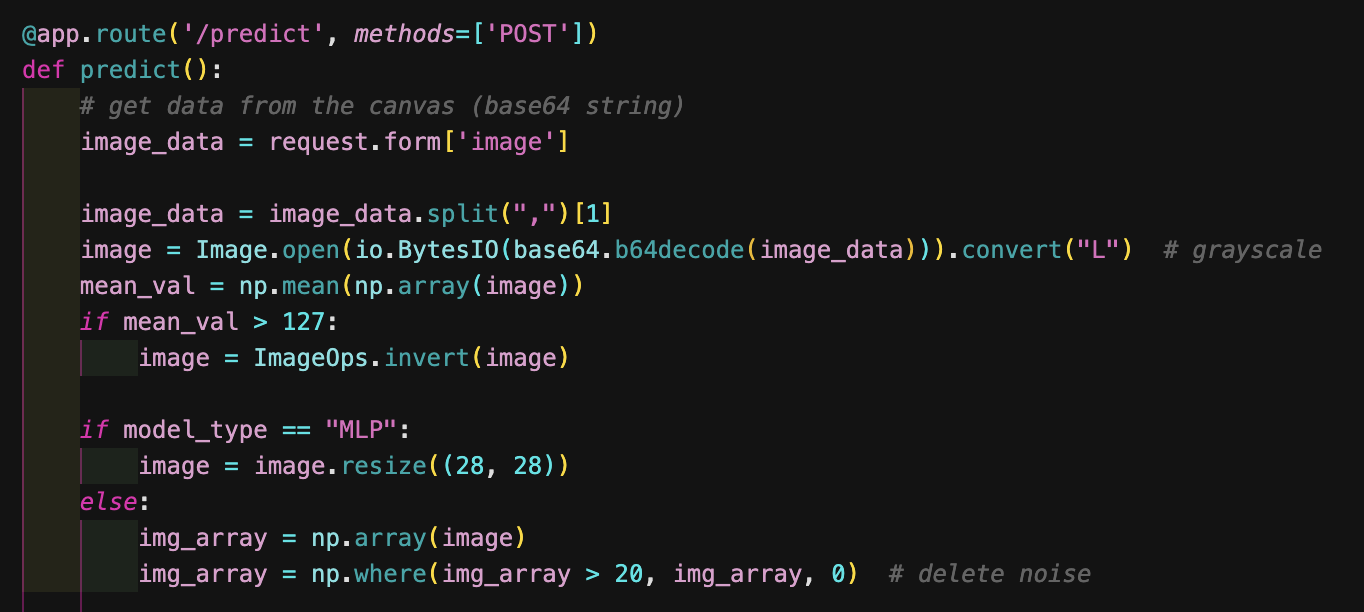


Figure Funktion predict(), get image and first preprocessing

A computer screen with text and numbers

AI-generated content may be incorrect.

Figure Funktion predict(), cut picture to bounding box

A screen shot of a computer program

AI-generated content may be incorrect.

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

A computer screen with text on it

AI-generated content may be incorrect.

Figure Funktion predict(), make predictions with model

## A.8 Funktion „def feedback()“

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

Figure Funktion feedback(), get data from template

A computer screen with text

AI-generated content may be incorrect.

Figure 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 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 Funktion create\_table()

A computer screen with text on it

AI-generated content may be incorrect.

Figure Funktion insert\_result()

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

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

A screenshot of a computer

AI-generated content may be incorrect.

Figure 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 Notebook, imports, parameters and load data

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Notebook, scale data

A black screen with white text

AI-generated content may be incorrect.

Figure Notebook, convert labels

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Notebook, transform x and define model MLP

A screenshot of a computer

AI-generated content may be incorrect.

Figure Notebook, compile and train model MLP

A screen shot of a computer program

AI-generated content may be incorrect.

Figure Notebook, evaluate and save model MLP

A screen shot of a computer program

AI-generated content may be incorrect.

Figure Notebook, build model CNN

A screenshot of a computer

AI-generated content may be incorrect.

Figure Notebook, compile and train model CNN

A screen shot of a computer code

AI-generated content may be incorrect.

Figure Notebook, evaluate and save model CNN

A screen shot of a computer program

AI-generated content may be incorrect.

Figure Notebook, define parameters to test

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure Notebook, define model, choose optimizer and train model

A screenshot of a computer program

AI-generated content may be incorrect.

Figure Notebook, start optuna study

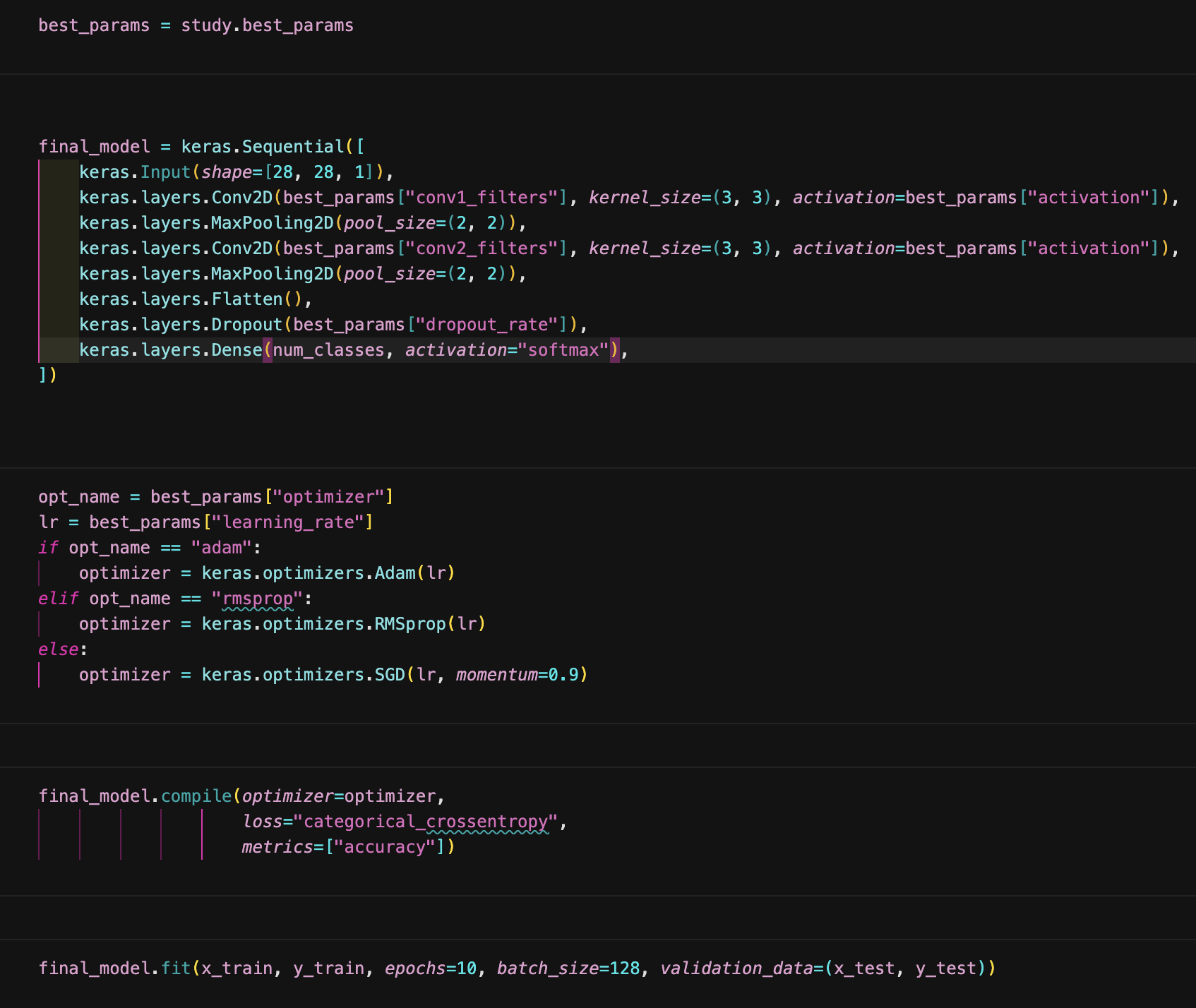


Figure Notebook, define and train final model

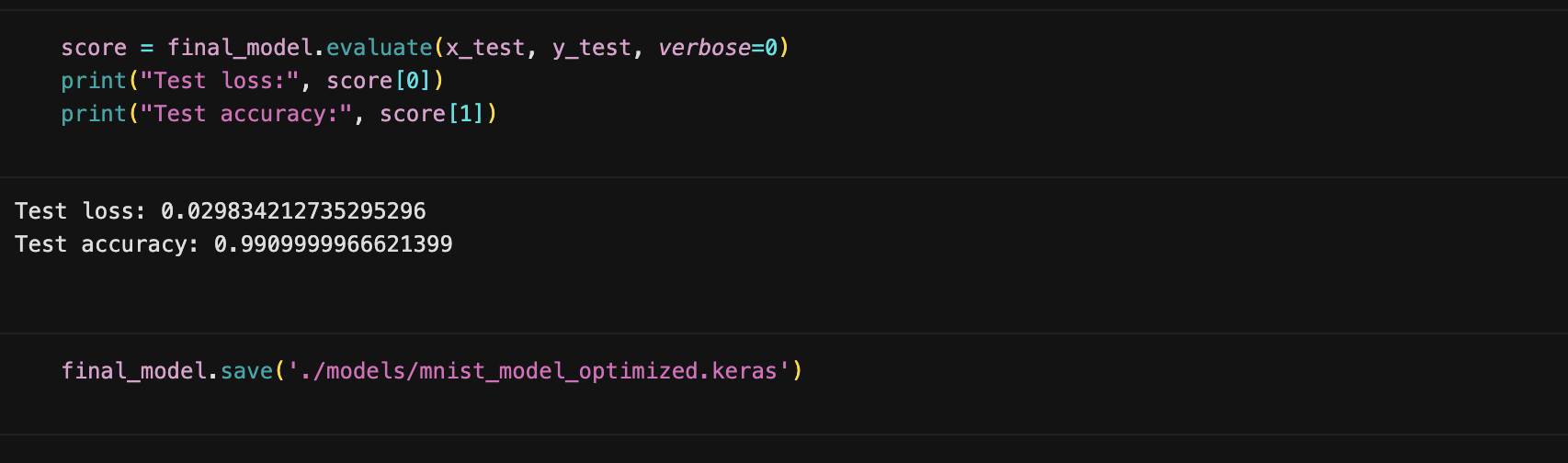
****

Figure Notebook, evaluate and save model CNN optimized

## 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 Git-Repository klonen

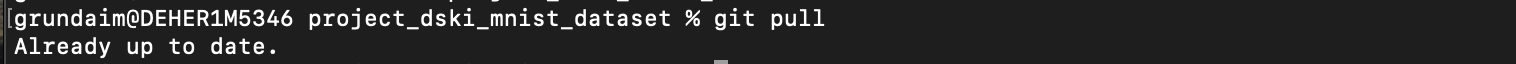


Figure pull-Request

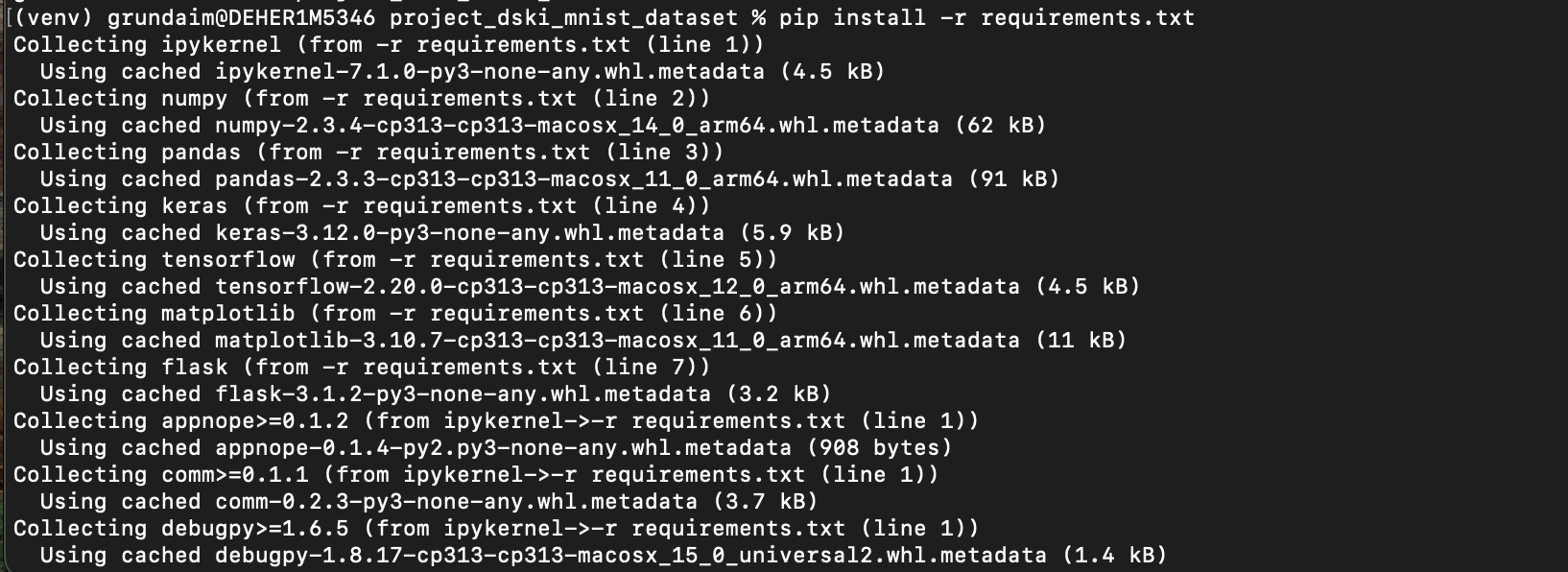


Figure Requirements installieren

A screen shot of a computer

AI-generated content may be incorrect.

Figure 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>