4 Science Challenge, 24. Wettbewerb

Aufgabe 3: Zahlen erkennen durch eine selbsttrainierte künstliche Intelligenz

Diese Aufgabe wird vom Institut für Mess- und Regelungstechnik der Fakultät für Maschinenbau gestellt. Falls ihr Interesse am Studiengang Maschinenbau habt, findet ihr hier weitere Informationen: https://www.maschinenbau.uni-hannover.de

Im Rahmen dieser Aufgabe wird eine künstliche Intelligenz zur Detektion von Zahlen und mathematischen Operatoren eingesetzt. Ein Großteil der Aufgaben beinhaltet einen programmiertechnischen Teil. Damit alle Aufgaben problemlos bearbeitet werden können, wird empfohlen, Google Colab (https://colab.research.google.com) zu verwenden.

Aufgabe 1 (8 Punkte) – Grundlagen künstliche Intelligenz

a) In diesem Aufgabenteil sollt ihr euch über die Begriffe der künstlichen Intelligenz informieren. Recherchiert die verschiedenen Arten von KI und welche Aufgaben sie lösen können. Beschreibt dafür kurz und knapp drei unterschiedliche Arten und welche Aufgaben diese lösen können. Nennt zudem fünf Bereiche, in denen KIs bereits eingesetzt werden. (4 Punkte)

b) In diesem Aufgabenteil sollt ihr euch insbesondere mit Bilderkennungsverfahren beschäftigen. Erforscht, wie sich Objekte auf Bildern erkennen lassen. Untersucht, anhand welcher Daten diese Verfahren trainiert werden und welche Auswirkungen die Daten auf das trainierte Modell haben. Beschreibt zudem beispielhaft, wie ein Verfahren zum Erkennen von Objekten trainiert werden kann. (4 Punkte)

Aufgabe 2 (22 Punkte) – Trainieren einer KI zum Erkennen von Zahlen und mathematischen Operatoren

In diesem Aufgabenteil werdet ihr eine KI so trainieren, dass diese in der Lage ist, Zahlen und mathematische Operatoren zu erkennen und zu lokalisieren. Abgegeben werden soll neben der schriftlichen Beantwortung der Fragen auch der verwendete Code (ihr könnt das Colab einfach herunterladen).

a) Die nachfolgenden Aufgaben enthalten alle einen Programmieranteil und sollen in Google Colab (https://colab.research.google.com) umgesetzt werden. Um euch mit diesem Werkzeug vertraut zu machen, sollt ihr in diesem Teil einfache Programmieraufgaben bearbeiten. Erstellt dafür zu Beginn ein neues Colab-Notebook. In diesem sollt ihr das Ergebnis der Rechnung 4+7 mit Hilfe des print-Befehls ausgeben lassen. In einem zweiten Code-Block sollt ihr eine for-Schleife erstellen, die bei 0 startet und bei 100 endet. In jedem Schritt sollen die Zahlen aufsummiert werden (Beispiel-Ausgabe: 0, 1, 3, 6, ...). (4 Punkte)

b) Erstellt nun ein neues Colab und richtet es gemäß den Hinweisen im Anhang ein. Ladet anschließend den bereitgestellten Datensatz 's4k_dataset.zip' herunter und fügt die .zip-Datei zu eurem Colab hinzu. Integriert auch die im Anhang definierten Funktionen in euer Colab. Startet das Training mit den in den Funktionen angegebenen Parametern. (2 Punkte)

c) Untersucht nach einem erfolgreichen Training die Ergebnisse, die in der Datei 'result.csv' festgehalten sind; diese befindet sich im automatisch erstellten Ordner 'train' bzw. 'train2' usw. Ladet dazu die

'result.csv' herunter und verwendet Excel (oder ein ähnliches Programm), um die Variable 'train/box_loss' aus einer der Spalten der CSV-Datei in einem Diagramm darzustellen. Beschreibt den Verlauf der Kurve und bezieht euch dabei auch auf das Wissen aus den ersten Aufgaben. (4 Punkte)

d) Um zu überprüfen, wie gut die KI das Erkennen von Handschriften gelernt hat, testet dies mit handschriftlichen Zetteln. Schreibt auf ein weißes Blatt Papier die Zahlen von 0 bis 9 sowie die mathematischen Operatoren +, -,*,/. Schreibt außerdem die beiden Gleichungen 1+5=9 und 3*6=18 auf separate Zettel. Ladet die drei Zettel in das Colab hoch. Achtet darauf, dass eure Bilder die Größe von 512 x 512 Pixel haben; schneidet gegebenenfalls die Bildbereiche mit den Zahlen aus dem größeren Bild heraus[1]. Überprüft mithilfe der model_prediction-Funktion, welche Zahlen und Operatoren auf den drei Bildern erkannt werden. Die Bilder mit den erkannten Zahlen und Symbolen werden unter /runs/predict/ abgespeichert. Falls eure Bilder nicht funktionieren, könnt ihr alternativ die Bilder aus dem Ordner s4k_dataset verwenden (z. B. Pfad zu dem Bild der Zahlenreihe: /content/s4k_dataset/test_bilder/zahlen_reihnfolge.png). (4 Punkte)

e) Beschreibt, was auf euren Bildern erkannt wird; vergleicht dazu eure Bilder mit denen aus dem Trainingsdatensatz (/content/s4k_dataset/train/images). Welche Zahlen werden bei euch erkannt? Sehen die Zahlen ähnlich aus wie in dem Testdatensatz? Vergleicht die Größe der Darstellung der Zahlen im Testdatensatz mit der Größe der Zahlen in euren Bildern. (4 Punkte)

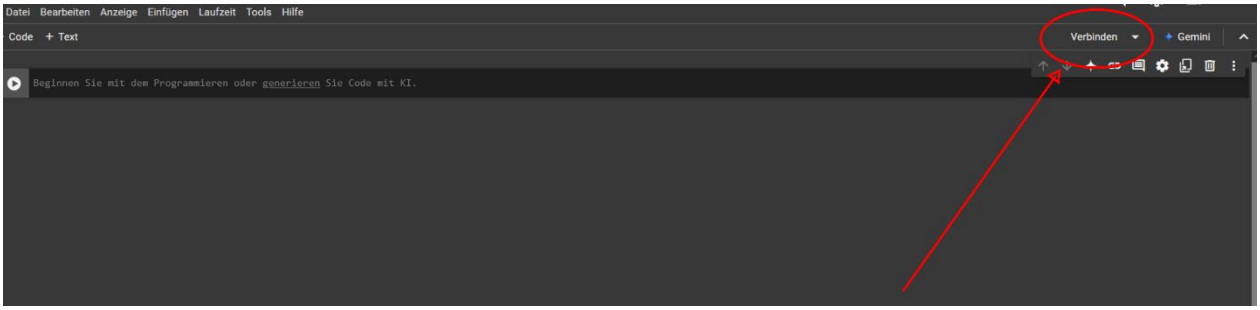
f) Passt die Funktion check_if_calculation_is_possible im Anhang so an, dass sie die beiden Gleichungen aus Aufgabenteil b) überprüfen kann. Beschreibt zunächst euer geplantes Vorgehen. Versucht anschließend, dies mithilfe von Python umzusetzen. Hinweis: Sortiert die erkannten Zahlen und mathematischen Operatoren nach ihrer Position auf dem Bild. Überprüft anschließend basierend auf der Position, wie die Gleichung formuliert ist. Ermittelt daraufhin, ob die Gleichung korrekt ist. (4 Punkte)

[1] https://www.iloveimg.com/crop-image

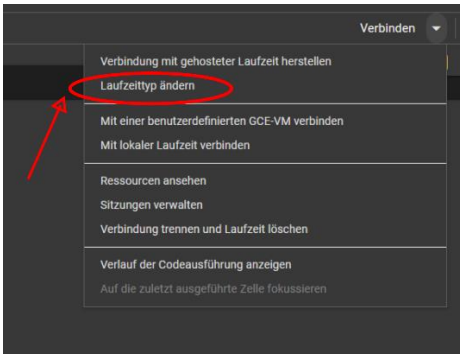
Code-Auszüge zu Aufgabenteil 2:

In diesem Aufgabenteil soll, wie in der Aufgabenstellung erwähnt, eine KI auf das Erkennen von Zahlen und mathematischen Operatoren trainiert werden. Auch hier könnt ihr wie im vorherigen Teil die Codeblöcke in euer Colab kopieren und einzeln ausführen. Für das Trainieren einer künstlichen Intelligenz sind leistungsstarke Computer notwendig. Das Colab-Tool bietet die Möglichkeit, weitere Rechenressourcen freizuschalten. Im Folgenden wird gezeigt, wie sich diese freischalten lassen:

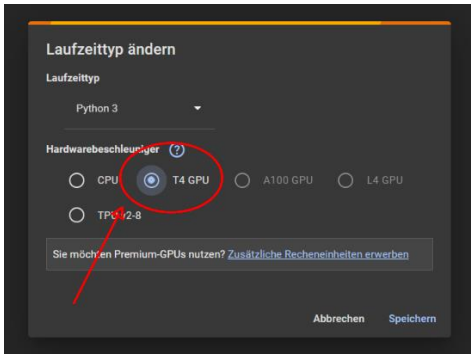
Schritt 1: Nach Öffnen eines neuen Colabs auf Verbinden drücken:



Schritt 2: Ändern des Laufzeittyps



Schritt 3: T4 GPU Auswählen



Führt diese Schritte erst aus, nachdem ihr das Colab entsprechend der gezeigten Vorgehensweise angepasst und die entsprechenden 's2k_dataset.zip'-Dateien in den Ordner geladen habt.

```
!pip install ultralytics
!unzip s4k_dataset.zip
```

Diese Zeile importiert das Python-Paket 'ultralytics'. Mit diesem werden wir eine KI darauf trainieren, Zahlen und mathematische Operatoren zu erkennen.

```
from ultralytics import YOLO
```

Diese Funktion trainiert die KI. Zu Beginn wird eine Struktur für die KI festgelegt. Wir verwenden hier die ‚yolo11m'-Struktur, eine mittelgroße Struktur. Die Größe der Struktur sollte in Abhängigkeit von der zu lösenden Aufgabe gewählt werden. In der Funktion werden auch das zu verwendende Optimierungsverfahren, die Lernrate und die Anzahl der zu trainierenden Epochen definiert. Pro Epoche wird einmal mit dem gesamten Datensatz trainiert; bei 50 Epochen wird also der Datensatz insgesamt 50-mal durchlaufen.

```
def model_training():
    model = YOLO('yolo11m.pt')
    model.train(data='/content/s4k_dataset/yolo_config.yaml',
                optimizer='SGD',
                project=r'/content',
                epochs=50,
                lr0=0.01,
                imgsz=512,
                device='0')
```

Aufrufen könnt ihr die Funktion mit dem nachfolgenden Block:

```
model_training()
```

Nach dem Ausführen des Blocks beginnt das Training, das in der Regel etwa 10 Minuten dauert. Nach jedem erfolgreichen Training werden die trainierten Netze in einem neuen Ordner (train_x) in eurem Colab abgespeichert. In diesem Ordner befinden sich alle Informationen zum Trainingsverlauf.

Zur Anwendung eurer trainierten KI könnt ihr die folgende Funktion verwenden.

```
def model_prediction(model_path, image_path):
    model = YOLO(model_path)
    results = model.predict(image_path,save=True)
    predictions = []
    for result in results:
        for box in result.boxes.data.tolist():
            x1, y1, x2, y2, confidence, class_id = box
            x_center = (x1 + x2) / 2
            y_center = (y1 + y2) / 2
            w = x2 - x1
            h = y2 - y1
            prediction = {
                'x_center': x_center,  # X-Koordinate des Mittelpunkts
                'y_center': y_center,  # Y-Koordinate des Mittelpunkts
                'w': w,  # Breite der Bounding Box
                'h': h,  # Höhe der Bounding Box
                'confidence': confidence,  # Konfidenzwert
                'class_id': int(class_id),  # Klassen-ID
```

```
                'class_name': model.names[int(class_id)]  # Klassenname
            }
            predictions.append(prediction)
    return predictions
```

Aufgerufen wird die Funktion wie folgt:

```
results = model_prediction(model_path =r'/content/train/weights/best.pt',
image_path='/content/s4k_dataset/test_bilder/zahlen_reihnfolge.png')
```

Hier müsst ihr nur die beiden Pfade model_path und image_path entsprechend eurer Daten anpassen. Die Funktion gibt alle erkannten Objekte (Zahlen und mathematische Operatoren) zurück, die in der Variablen results abgespeichert werden. Zudem wird ein Bild, auf dem die erkannten Objekte markiert sind, im Ordner runs/predictions_x abgespeichert.

Diese Funktion soll überprüfen, ob die handgeschriebenen Gleichungen korrekt sind. Dafür werden der Funktion die erkannten Zahlen und Operatoren übergeben. Die Logik hinter der Überprüfung soll von euch erarbeitet werden. Angefangen werden kann mit den Hinweisen, die in der Funktion stehen.

```
def check_if_calculation_is_possible(result):
    # Hier sollt ihr Code schreiben. Es soll überprüft werden, ob die Glei-
chungen korrekt sind.
    # Sortiert dabei die erkannten Objekte nach den Mittelpunkten der
Bounding Boxen.
    # Geht dann von links nach rechts durch und checkt das Ergebnis.
    # Beispielcode zum Durchgehen der Ergebnisse:
    for result in results:
        print(result['class_name']) # über 'class_name' kann auf den Namen
der Klasse zugegriffen werden
        # Über 'x_center', 'y_center' usw. auf die anderen Werte
        # z.B. result['x_center'] auf den x-Wert des Mittelpunkts
```

Aufgerufen wird die Funktion wie folgt:

```
check_if_calculation_is_possible(results)
```

Dabei sind results die Ergebnisse aus der Funktion model_prediction. Nach der Überprüfung soll ausgegeben werden, ob die Gleichung richtig ist.

Viel Erfolg bei der dritten Aufgabe!

Allgemeine Hinweise
Einsendeschluss: Sonntag, 05. Januar 2025, 19:59 Uhr

Gebt eure Lösungen über Stud.IP ab: https://studip.uni-hannover.de

Das zulässige Dateiformat für die zusammengeschriebene Lösung (mit eingebetteten Bildern) ist PDF. Bitte ladet eure Dateien rechtzeitig hoch.
Gebt innerhalb der Datei euren Teamnamen, die Namen der Teammitglieder sowie deren Schulen an. Benennt eure Datei nach folgendem Schema: „Teamname_Aufgabe3".

Das Hochladen funktioniert wie folgt:
Loggt euch mit den bei eurer Anmeldung zur 4 Science Challenge angelegten Zugangsdaten auf der Stud.IP-Seite ein (bitte nutzt dazu den „Login ohne WebSSO"). Geht dann auf „Meine Veranstaltungen" und auf die 4 Science Challenge 2024/2025. Geht dann oben auf „Dateien" und auf den Ordner „Upload Aufgabe 3". Dort könnt ihr entweder über „Dokument hinzufügen" oder über „Dateien hochladen" eure Lösungsdatei hochladen.

Wenn ihr die Datei hochgeladen habt, öffnet sich ein Fenster, in dem u. a. nach Lizenzinformationen gefragt wird. Dieses braucht ihr nicht weiter zu beachten und könnt einfach auf „Speichern" klicken. Bitte achtet darauf, dass ihr eure Dateien wirklich innerhalb des Ordners „Upload Aufgabe 1" hochladet und nicht außerhalb davon, da ansonsten die anderen Teams eure Dateien sehen können.

Die Teilnahmebedingungen und weitere Informationen findet ihr unter www.uni-hannover.de/4sciencechallenge

Der Rechtsweg ist ausgeschlossen.