Aufgabenstellung

Ziel war es, ein vollständiges System zur Klassifikation von Straßenschildern zu entwickeln, das aus drei zentralen Bausteinen besteht:

1. Vorverarbeitung von Bilddaten
2. Training eines robusten Klassifikationsmodells
3. Bereitstellung einer interaktiven Web-App zur Bildvorhersage

1. Datenvorverarbeitung

Die Bilddaten bestanden aus 211 Klassen mit bis zu 500 Bildern pro Klasse, jeweils mit optionalen JSON-Metadaten (https://synset.de/datasets/synset-signset-ger/). Hier werden die Bilder einheitlich vorverarbeitet und in Trainings-, Validierungs- und Testdatensätze aufgeteilt:

- Nutzung von Albumentations für Resize, Padding & Normalisierung
- Einheitliches Format (224×224 RGB)
- Stratified Split mit sklearn zur Klassenerhaltung
- Speichern der Daten als .npy und .pkl

2. Modelltraining mit PyTorch

Ein ResNet18-Modell wurde verwendet, um aus den Bilddaten die Klassenzugehörigkeit vorherzusagen.

Gelöst durch:

- Implementierung eines Trainingsskripts mit konfigurierbaren Hyperparametern
- Nutzung von torchvision.models.resnet18(pretrained=True) mit Custom-Classifier
- Training mit CrossEntropyLoss & Adam
- Speichern des Modells (state_dict) als .pth

- Ergebnisse wurden verglichen und dokumentiert (Accuracy, Loss, etc.)

3. Streamlit Web-App

Ziel: Eine einfache Benutzeroberfläche, in der ein Nutzer ein Bild hochlädt und sofort eine Vorhersage bekommt:

- Erstellung eines app.py mit streamlit
- Einbindung der Modellvorhersage über utils.py
- Anzeige der Vorhersage und der hochgeladenen Bildvorschau
- Zusatz: Timer, der misst, wie lange die Vorhersage dauert

Die App lädt das Modell (.pth) und den label_encoder.pkl, bereitet das Bild vor, führt eine Vorhersage durch und zeigt das Ergebnis benutzerfreundlich an.

Testing

Ein separates Skript test_preprocessing.py stellt sicher, dass:

- Bilder korrekt geladen und transformiert wurden
- Die Shapes und Labelverteilungen stimmen
- Keine Fehler beim Encoding auftreten

Deployment & Reproduzierbarkeit

- Projektstruktur ist modular aufgebaut
- Alle Schritte (Preprocessing, Training, App) sind als separate Skripte nutzbar
- Das Modell kann später auf einen Mikroprozessor für das autonome Fahren implementiert