UE 4 – Mehrklassen-Klassifikation mit Neuronalen Netzen

Überblick

In dieser Übungseinheit machen wir uns die Tensorflow-Open-Source-Plattform für maschinelles Lernen zunutze, um die Klassifikation von 43 Verkehrsschildklassen des GTSRB-Datensatzes durchzuführen. Wir werden mit der benutzerfreundlichen sequentiellen Keras-API arbeiten.

Die Tensorflow-Tutorials für Anfänger https://www.tensorflow.org/tutorials?hl=de könnt ihr als Einstieg in die Keras Grundlagen nutzen.

23.06.2022 – Vorbereitung

Das Ziel der Vorbereitungsaufgabe ist, dass ihr euch mit den 4 Schritten einer Modellentwicklung mit Keras vertraut macht:

1. Aufbau des Modells
2. Kompilieren des Modells
3. Training des Modells
4. Evaluation des trainierten Modells

Folgende Fragen solltet ihr dabei beantwortet haben:

- Wie erfolgt die Datenaufbereitung („Loading & Preprocessing“)?
- Was ist ein Layer?
- Was ist ein Dense-Layer?
- Was ist ein Sequential model?
- Wofür werden folgende Befehle / Funktionen benutzt, welche Parameter müssen für die Verwendung dieser Befehle in Abhängigkeit von der Aufgabenstellung und den verwendeten Daten festgelegt werden:
    - Model()
    - Sequential()
    - keras.layers.Dense()
    - add()
    - compile()
    - fit()
    - evaluate()
    - predict()

Als Informationsquelle bietet sich auch https://keras.io/ an. Gerne könnt ihr euch alles anschauen, fokussiert euch in diesem Schritt primär auf die Grundlagen:

- https://keras.io/about/
- https://keras.io/getting_started/intro_to_keras_for_engineers/

- https://keras.io/guides/sequential_model/
- https://keras.io/guides/training_with_built_in_methods/

Als Zusatzmaterialien könnt ich auch das **Keras_Cheat_Sheet_Python.pdf** sowie die Lernressourcen von Keras-Entwicklern nutzen.

Praktische Anwendung

Um das erlernte praktisch anzuwenden, könnt ihr den first end-to-end example mit den HOG-Features aus der dritten Übung als Inputs nachimplementieren und die Performance eures Neuronalen-Netz-Modells mit der Performance bspw. des SVM-Modells vergleichen.

30.06.2022 – Aufgabe

In dieser Aufgabe werdet ihr eine andere Netzarchitektur – *Convolutional Neural Networks* (ConvNets) – kennenlernen. Diese Netzarchitektur wurde speziell für die Verarbeitung von Bildern konzipiert. Zusätzlich zu den Dense-Layers, die für die Klassifikationsaufgabe zuständig sind, kommen zwei weitere ConvNets Layer-Typen, die die Merkmalsextraktionsaufgabe übernehmen. Das sind Convolutional Layers und Pooling Layers.

Eure Aufgabe besteht darin, ein simples ConvNet-Modell zur Klassifikation von Verkehrszeichen zu implementieren, zu trainieren und mit euren Beispielbildern zu testen. Diese Aufgabe könnt ihr angelehnt an die Schritte der Modellentwicklung mit Keras in folgende Teilaufgaben unterteilen:

1. Datenaufbereitung
2. Aufbau des Modells
3. Kompilieren des Modells
4. Training des Modells
5. Evaluation des trainierten Modells
6. Speichern des Modells
7. Verwendung des gespeicherten Modells zur Klassifikation eigener Beispielbilder

Die angelegte Jupyter-Notebook-Vorlage enthält Hinweise zu den einzelnen Schritten.

Vergleicht die Performance eures ConvNet-Modells mit der Performance des Neuronalen-Netz-Modells mit den HOG-Features. Welches Modell schneidet besser ab? Habt ihr eine Idee, woran es liegen könnte?
比较你的ConvNet模型和带有HOG特征的神经网络模型的性能。哪种模式表现得更好？你有什么想法吗，可能是什么原因？