SketchMe Backend Dokumentation

Abhängigkeiten

Damit das Backend lauffähig ist, werden folgende Abhängigkeiten benötigt:

- Python3
- Tensorflow

Low-Level-Framework, welches von Keras benutzt wird. Tensorflow wird von Keras benötigt, um dessen Machine-Learning zu implementieren.

- Keras

High-Level-Framework, welches Machine-Learning Algorythmen implementiert, welche von SketchMe benutzt werden.

- h5py

Von Keras benutzt, um das trainierte Modell zu speichern und zu laden.

- numpy

Library zur Berechnung von mehrdimensionalen Vektoren und Matrizen. Wird von SketchMe benötigt zur Verarbeitung der Bilddaten.

Modell

Modell speichern

Um ein vorher erstelltes und trainiertes Modell zu speichern, benutzen wir: `model.Save_Model();`. Das gespeicherte Modell befindet sich im selben Ordner unter dem Namen: `SketchMe.hdf5`

Modell laden

Falls im selben Ordner ein Modell Namens `SketchMe.hdf5` existiert, wird es mit `model.Load_Model();` in das Backend geladen.

Trainingsphase

Trainingsdaten

Um unser Modell zu trainieren, bedienen wir uns der numpy Bitmaps von Googles “Quick, draw!”.

Um alle Daten runterzuladen, haben wir ein Shellskript `categories.sh` geschrienen. Dieses lädt die Bilddaten aller Kategorien auf den eigenen Rechner. Die Trainingsdaten sind für jede Kategorie als 28x28 bitmap in einem eigenem numpy Array.

Vorbereitung der Daten

Damit unser Modell korrekt Trainiert werden kann, sind zum Schluss gekommen, dass es am besten ist, wenn die Daten zufällig (in Reihenfolge, als auch Kategorie) eingespeist werden. Um dies zu bewerkstelligen, haben wir ein Python Programm geschrieben: `createData.py`. Dieses liest aus der Datei `categories.txt` welche Kategorien in die Trainingsdaten einbezogen werden und erstellt aus diesen mehrere Dateien.

- `x_test.h5`: Test- bzw. Validierungsdaten
- `x_train.h5`: Trainingsdaten
- `y_test.h5`: Die Kategorien, repräsentiert als Zahl, der Test- bzw. Validierungsdaten
- `y_train.h5`: Die Kategorien, repräsentiert als Zahl, der Trainingsdaten

Insgesammt werden aus jeder Kategorie, welche angegeben wurde, 1000 Bilder genommen. Anschließend werden die Bilder gemischt und 80% werden als Trainingsdaten verwendet. Die restlichen 20% werden für Test bzw. Validierung verwendet.

Training

Bevor wir das Modell trainieren können, müssen die zuvor präparierten Datensätze eingelesen und in die korrekte "Listendarstellung" gesetzt werden. Hierzu werden die Methoden `Load_Data();` und `Format_Data();` benutzt.

Sind die Daten eingelesen und ein Modell erstellt bzw. geladen, kann man der Funktion `Train_Model();` sein Modell trainieren. Eingestellt sind 30 Epochen d.h. 30 Trainingseinheiten mit dem gleichem Datensatz. Hier werden aus dem Trainings-Datensatz 75% der Daten für das eigendliche Training genutzt und die letzten 25% zur Validierung nach jeder Epoche.

Da 30 Epochen jedoch viel sein kann, benutzen wir ein von Keras bereitgestelltes Callback: `EarlyStopping()` . Dieses Überwacht den Lernfortschritt und bricht das Training ab, wenn sich über 3 Epochen keine verbesserung im Training zeigt.

Auch verringern wir die Lernrate des Trainings wenn es stagniert. Dies wird durch den Callback `ReduceLROnPlateau()` ermöglicht.

Validierung

Die Validierung des Trainings kann man mit dem Test-Datensatz ausführen. Hierfür müssen die daten wie für das Training eingelesen werden. Ist dies getan, so kann man die Validierung mittels `Evaluate_Model();` starten.

Benutzung

Damit unser Modell Bilder erkennen kann, muss man, nachdem man es geladen hat, die Methode `Predict(bitmap)` ausführen. Diese akzeptiert eine 28x28 Bitmap und gibt einen Vektor zurück. Dieser Vektor enthällt die Wahrscheinlichkeit für jede Kategorie, dieser anzugehören. Hier gibt die Summe aller Wahrscheinlichkeiten 100%.