TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Projektarbeit

Nutzerdokumentation ML.NET-Bildklassifzierung

Konsolenanwendung

Alexander Kaufmann
Robert Mende

Nanotechnologie
Chemie

27. Juli 2021

Inhaltsverzeichnis

1 Motivation

Machine Learning stellt eine Möglichkeit dar, Computer analog zum menschlichen Lernen anhand von Erfahrungen lernen zu lassen. Dabei umfasst das Machine Learning ein weites Feld möglicher Anwendungsbereiche, wobei die Bildklassifizierung ein solches darstellt. Mit der vorliegenden Software können Bilder mittels eines zuvor in der Software trainierten Modells klassifiziert werden. Zum Einsatz kommt dazu ein von Google erstelltes neuronales Netz „Inception v1“ nach dem Vorbild des Dokumentationsbeispieles von ML.Net.

2 Anwendungsfeld der Software

Die Software kann prinzipiell zur Kategorisierung von Bildern der Typen .jpg und .png verwendet werden. Die Verwendung der mit der Software erzeugten Vorhersagemodelle ist jedoch nur mit ungelabelten Bildern sinnvoll, die tatsächlich zu einem der vorher trainierten Labels gehören. Dementsprechend führen beispielsweise Bilder von Fahrzeugen bei einem Modell, das zur Klassifizierung verschiedener Blumen trainiert wurde, zu willkürlichen Zuweisungen. Im Gegenzug kann die Software je nach Größe der Trainingsdaten, Unterschiedlichkeit zwischen den trainierten Labels etc. durchaus zuverlässig neue Bilder klassifizieren. Weiterhin kann das Modell vom Nutzer auf neue Kategorien trainiert werden. Dazu kommt das Neurale Netz `InceptionV1` von Google zum Einsatz. Die notwendigen Bilder werden aus der freien Datenbank Open Images Dataset V6 heruntergeladen. Auf Basis eines neu trainierten Modells kann die Bildklassifikation wieder aufgerufen werden.

3 Benutzung der Software

3.1 Vorbereitungen

3.1.1 Releasebuild

Runtime

Die Software wurde für `.NET 5.0` geschrieben. Zu Ausführung ist die entsprechende Runtime notwendig.

Zusätzliche Dateien

Zum Trainieren neuer Modelle können Bilder aus einem AWS Bucket heruntergeladen werden. Der Datenbankindex ist jedoch zu groß für Github und muss separat von Hand heruntergeladen werden. Datei besitzt eine Größe von ca. 2,5 Gb. Nach dem Herunterladen muss die Datei auf eine Ebene mit der `.Index`-Datei kopiert werden und in `imageIDs.csv` umbenannt werden. Die Ordnerstruktur sollte Abb. 1 entsprechen. Das Programm ist nun einsatzbereit.

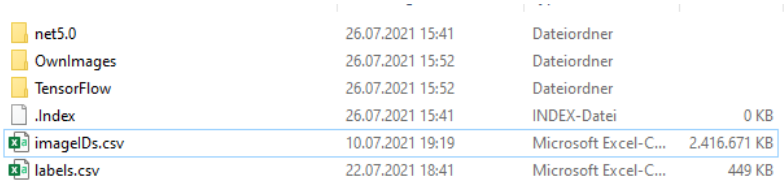


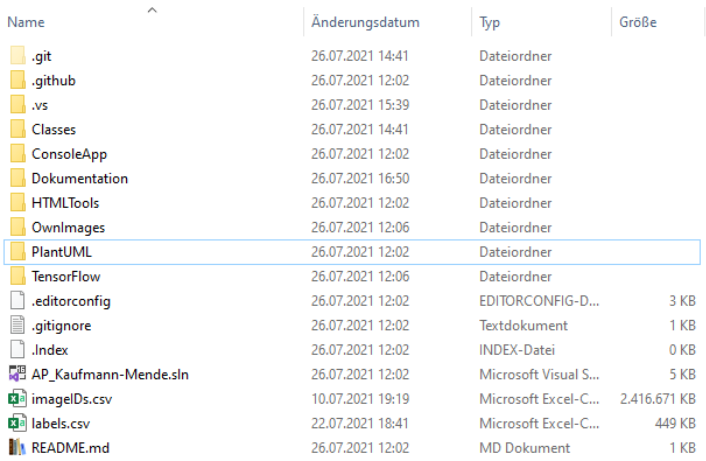
Abb. 1: Ordnerstruktur für den Release-Build

3.1.2 Build von Sourcecode

Die Quellcode ist über Github download verfügbar und kann über die `.Net-SDK` kompiliert werden. Die zusätzliche `.csv`-Datei ist analog dem Releasebuild herunterzuladen und auf einer Ebene mit der `.Index`-Datei zu speichern. Das Programm kann durch das erstellen eines Builds der `ConsoleApp` gestartet werden. Die Ordnerstruktur sollte Abb. 2 entsprechen. Wenn der Ordner `TensorFlow` fehlt, muss dieser von Hand erstellt werden und das neuronale Netz von Hand direkt dort eingefügt werden (`.pb-Datei`). Sollte die vorhandene `tensorflow_inception_graph.pb`-Datei im Ordner `TensorFlow` korumpiert sein, befindet sich im `zip`-Verzeichnis `inception5h.bac` ein Backup.

NuGet-Packages

Sämtliche benötigten NuGet-Packages werden in der Solution- bzw. Projekt-Datei vermerkt. Die benötigten Pakete müssen im Paketmanager mit `nuget restore AP_Kaufmann-Mende.sln` wieder hergestellt werden, **sollten sie nicht automatisch durch NuGeT heruntergeladen werden**. Achtung: es werden ca. **2 Gb** des in der `.NET-SDK` hinterlegten Nutzerverzeichnisses zusätzlich belegt. Hiermit sind die Vorbereitungen abgeschlossen.



Name	Änderungsdatum	Typ	Größe
.git	26.07.2021 14:41	Dateiordner	
.github	26.07.2021 12:02	Dateiordner	
.vs	26.07.2021 15:39	Dateiordner	
Classes	26.07.2021 14:41	Dateiordner	
ConsoleApp	26.07.2021 12:02	Dateiordner	
Dokumentation	26.07.2021 16:50	Dateiordner	
HTMLTools	26.07.2021 12:02	Dateiordner	
OwnImages	26.07.2021 12:06	Dateiordner	
PlantUML	26.07.2021 12:02	Dateiordner	
TensorFlow	26.07.2021 12:06	Dateiordner	
.editorconfig	26.07.2021 12:02	EDITORCONFIG-D...	3 KB
.gitignore	26.07.2021 12:02	Textdokument	1 KB
.Index	26.07.2021 12:02	INDEX-Datei	0 KB
AP_Kaufmann-Mende.sln	26.07.2021 12:02	Microsoft Visual S...	5 KB
imageIDs.csv	10.07.2021 19:19	Microsoft Excel-C...	2.416.671 KB
labels.csv	22.07.2021 18:41	Microsoft Excel-C...	449 KB
README.md	26.07.2021 12:02	MD Dokument	1 KB

Abb. 2: Ordnerstruktur des Projektes, fertig zum kompilieren

4 Nutzung der Software

4.1 Benutzeroberfläche

Ausgeführt wird eine Konsolenapp in `.NET 5`, die dem Nutzer nach Start der Software zwei Möglichkeiten bietet:

- Das Training eines Modells, um nachfolgend damit Bilder klassifizieren zu können
- Die Klassifizierung von Bildern, die sich in einem vorgegebenen Ordner befinden

Die Entscheidung ist anhand der Eingabe der Taste 1 beziehungsweise 2 unmittelbar nach dem Start zu treffen.

4.1.1 Auswahl des Trainingsmodus

Der Trainingsmodus dient programmintern zur Erstellung eines Modells, das Bilder im vom Nutzer vorgegebene Kategorien einteilt. Dementsprechend wird der Nutzer nach Eintritt in den Trainingsmodus aufgefordert, einen Suchbegriff vorzugeben. Anhand dieses Begriffes wird in der Datenbank nach Labels gesucht, die den vorgegebenen Begriff enthalten. Diese Labels werden dem Nutzer anschließend zum Training vorgeschlagen.

Der Nutzer wählt anschließend über die Eingabe einer oder mehrerer Zahlen diejenigen Labels, auf die er sein Modell trainieren möchte. An dieser Stelle ist die Interaktion des Nutzers mit dem Trainingsmodus abgeschlossen und das Modell wird trainiert. Die Bilder werden zuvor aus einem Amazon Cloud Storage (AWS Bucket) heruntergeladen. Je nach Umfang der Trainingsbilder kann dieser Vorgang viel Zeit in Anspruch nehmen.

4.2 Auswahl des Klassifizierungsmodus

Im Klassifizierungsmodus können anhand eines zuvor erstellten Modells Bilder aus dem Ordner `OwnImages`, der sich im gleichen Ordner wie die `.Index` oder die `AP_Kaufmann-Mende.sln`-Datei befindet, geladen und klassifiziert werden. Die Bilder sollten dabei im `.jpg`- oder `.png`-Format vorliegen.

Wird der Klassifizierungsmodus im Hauptmenü durch Drücken der Taste 1 ausgewählt, so werden alle indizierten, zuvor trainierten Modelle angezeigt. Über Eingabe von deren Index

und die Bestätigung mit Enter kann nun ein einzelnes Modell geladen werden. Dieses Modell wird anschließend genutzt, um die im `OwnImages`-Ordner liegenden Bilder zu klassifizieren.

Die Ausgabe der vorhergesagten Labels erfolgt anhand einer `HTML`-Datei im Ordner `OwnImages`. In der `HTML`-Datei befinden sich die Label, welche Verlinkungen auf die jeweiligen Bilder besitzen.

5 Abbruchkriterien

Im Falle verschiedener Fehler wird das Programm beendet:

Kategorisierung eigener Bilder

- Kein trainiertes Modell vorhanden

Trainieren eigener Modelle

- `csv`-Dateien nicht vorhanden oder der Header der `csv`-Dateien ist falsch
- keine Verbindung zu Amazon-Diensten möglich
- Tensorflow-Inception-Neural-Network nicht vorhanden

Außerdem bricht das Programm ab, wenn nur unzureichende Schreib-/Leserechte vorliegen.

6 Known Issues

Die Download-API von AWS friert teilweise das Programm ein, in dem Falle muss das Programm neugestartet werden. Leider konnten dazu keine weiteren Information gefunden werden.

7 Umsetzung

Die Programmierung erfolgte von Alexander Kaufmann und Robert Mende. Die Grundstruktur des Training des Tensorflow-Abschnittes stammt aus Dokumentationsbeispiel von ML.Net

8 Links

- Dokumentationsbeispiel ML.Net Bildklassifizierung: https://docs.microsoft.com/de-de/dotnet/machine-learning/tutorials/image-classification
- Open Images Dataset V6 https://storage.googleapis.com/openimages/web/index.html
- Microsoft `.Net` https://dotnet.microsoft.com/download
- Image-IDs der Open Images Dataset V6-Datenbank https://storage.googleapis.com/openimages/v6/oidv6-train-annotations-human-imagelabels.csv
- neuronales Netz Inceptionv1 https://storage.googleapis.com/download.tensorflow.org/models/inception5h.zip

Anhang

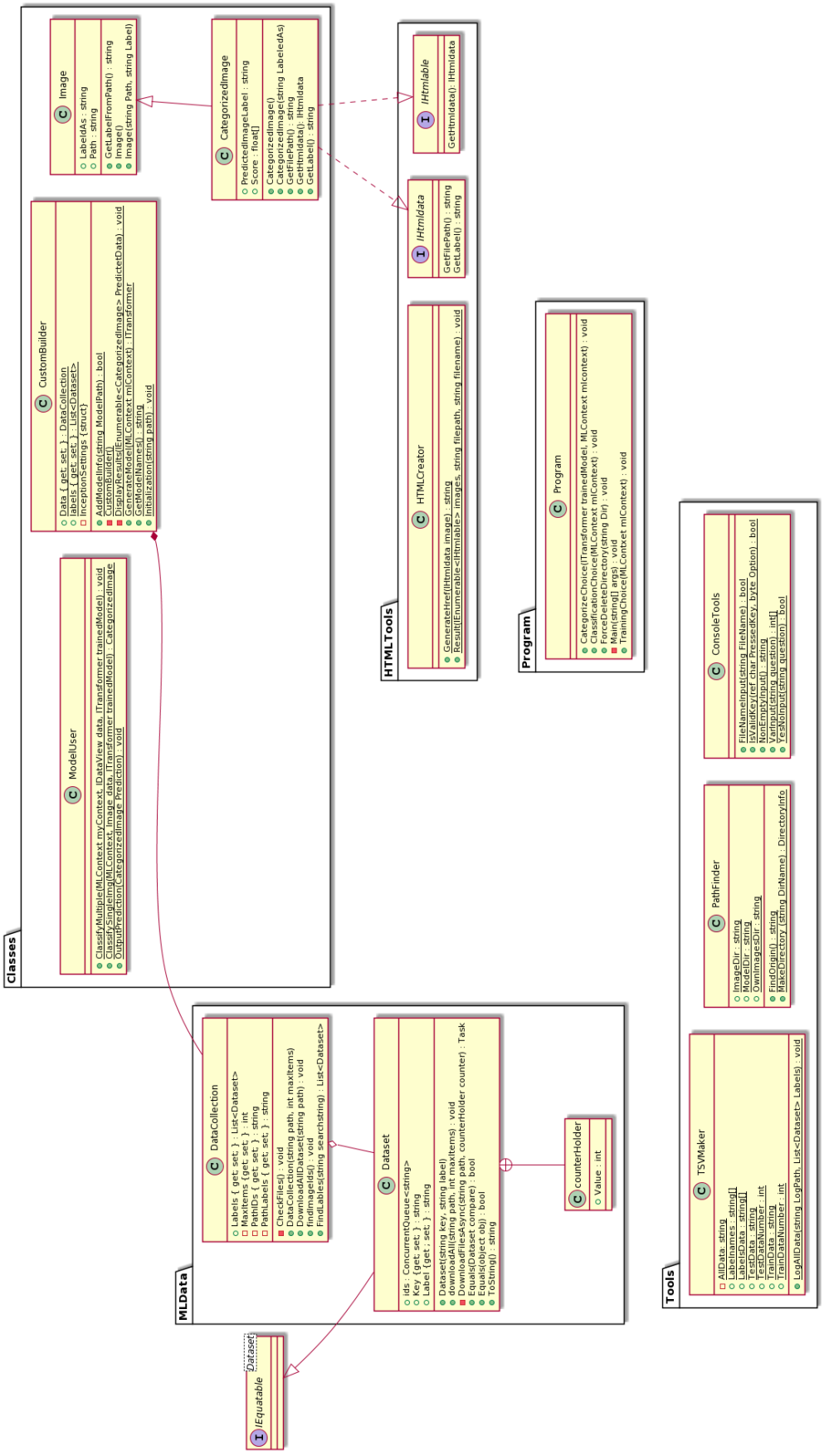


Abb. 3: Klassendiagramm des finalen Programms