



TECHNISCHE UNIVERSITÄT  
BERGAKADEMIE FREIBERG

Die Ressourcenuniversität. Seit 1765.

**Projektarbeit**

# **Nutzerdokumentation ML.NET-Bildklassifizierung**

**Konsolenanwendung**

**Alexander Kaufmann  
Robert Mende**

Nanotechnologie  
Chemie

27. Juli 2021

## Inhaltsverzeichnis

<b>1 Motivation</b>	<b>2</b>
<b>2 Anwendungsfeld der Software</b>	<b>2</b>
<b>3 Benutzung der Software</b>	<b>3</b>
3.1 Vorbereitungen . . . . .	3
3.1.1 Releasebuild . . . . .	3
3.1.2 Build von Sourcecode . . . . .	3
3.1.3 Vorbereitungen in Visual Studio . . . . .	3
<b>4 Nutzung der Software</b>	<b>4</b>
4.1 Benutzeroberfläche . . . . .	4
4.1.1 Auswahl des Trainingsmodus . . . . .	5
4.2 Auswahl des Klassifizierungsmodus . . . . .	5
<b>5 Abbruchkriterien</b>	<b>5</b>
<b>6 Known Issues</b>	<b>6</b>
<b>7 Umsetzung</b>	<b>6</b>
7.1 Download der Bilder . . . . .	6
<b>8 Links</b>	<b>6</b>
<b>Anhang</b>	<b>8</b>

## 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 [Dokumentationsbeispiels](#) 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 sich tatsächlich einem der vorher trainierten Labels zuordnen lassen. 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

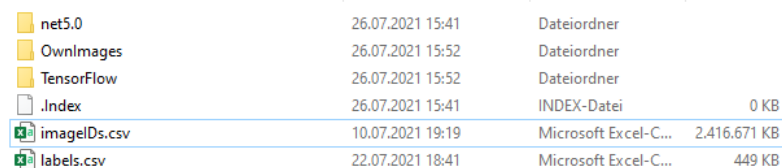
#### 3.1.1 Releasebuild

##### 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.



net5.0	26.07.2021 15:41	Dateiordner	
OwnImages	26.07.2021 15:52	Dateiordner	
TensorFlow	26.07.2021 15:52	Dateiordner	
.Index	26.07.2021 15:41	INDEX-Datei	0 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

**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.

#### 3.1.3 Vorbereitungen in Visual Studio

Da zur Erstellung der Software Visual Studio genutzt wurde, wird empfohlen, die Kompilierung des Quellcodes ebenfalls in jener IDE durchzuführen. In Visual Studio wird zunächst

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
imageDs.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

das Repository geklont. Daraufhin wird das geklonte Repository geöffnet und am rechten Bildschirmrand erscheint der Projektmappenexplorer. Durch einen Doppelklick auf `AP_Kaufmann-Mende.sln` wird die zugehörige Solution geöffnet. Hier ist standardmäßig 'Classes' als Startprojekt festgelegt. Da dies jedoch eine Klassenbibliothek darstellt, ist stattdessen 'ConsoleApp' als Startprojekt festzulegen. Dies geschieht durch Rechtsklick auf das Projekt und anschließenden Klick auf die Fläche **Als Startprojekt festlegen**. Dies ist in Abb. 3 dargestellt.

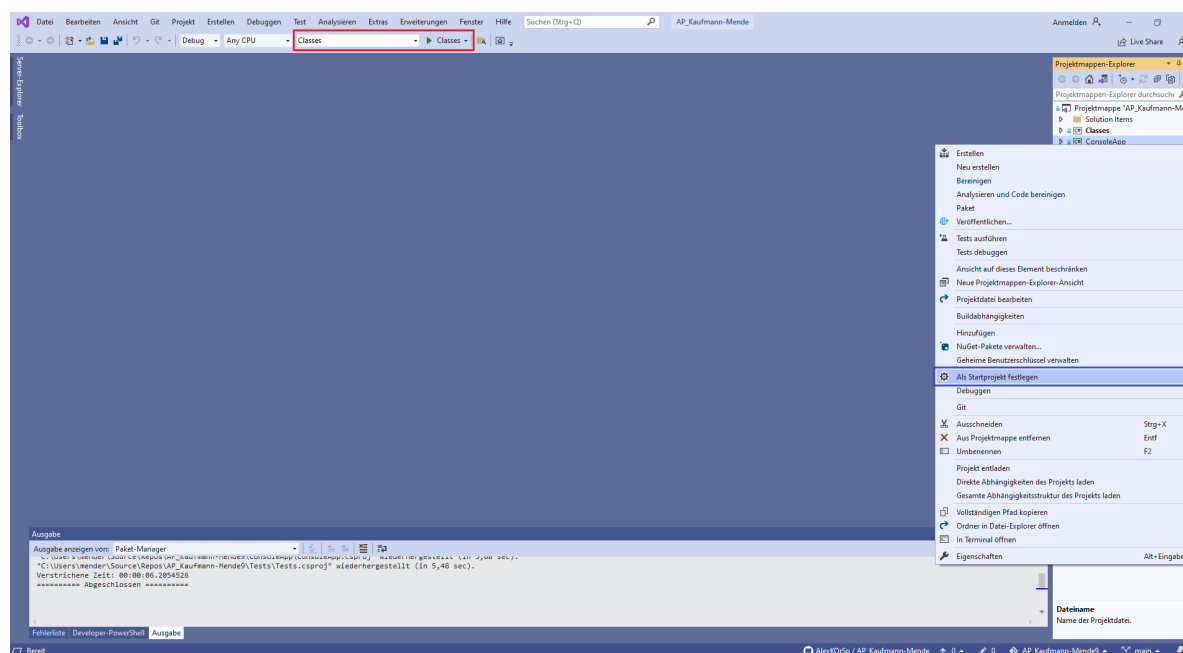


Abb. 3: Festlegen des Startprojektes in Visual Studio

## 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 in 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. Die einzelnen Funktionen sind mittels der integrierten XML-Dokumentation von Visual Studio kommentiert.

### 7.1 Download der Bilder

Die Bilder der verschiedenen Kategorien sind in einem AWS-Bucket gespeichert. Die Indizes aller Bilder mit den verknüpften Labels sind in der `csv`-Datei `imageIDs.csv` enthalten. Wenn sich der Nutzer für die entsprechende Kategorie entschieden hat, wird die Datei mit der Funktion `findImageIds()` in `DataCollection` für alle Zeilen durchlaufen und die entsprechenden Bilder-IDs mit dem richtigen Label werden in die Queue eines `Datasets` geschrieben (vgl. Abb. 4). Alle `Datasets` werden von einer `DataCollection` verwaltet. Nachdem `findImageIds()` abgeschlossen hat, wird in jedem `Dataset` der Reihe nach der Download gestartet. Übergeordnet ist dafür die Funktion `DownloadAllDataSets` in `Datacollection` verantwortlich. In jedem `Dataset` wird die Funktion `downloadAll()` aufgerufen. Diese erzeugt 5 asynchrone `Downloadtasks`. Die `AWSSDK` stellt dafür entsprechende asynchrone Methoden bereit. In der asynchronen Task wird ein `AmazonS3Client` erzeugt, welcher für die Nutzung der AWS-Dienste notwendig ist. Mit den entsprechenden Informationen des Buckets von Open Images kann ein `Transferclient` erzeugt werden, welcher die API für Uploads und Downloads zur Verfügung stellt. Über diesen werden die Bilder dequeued.

Zwischen allen Tasks wird eine Counter-Instanz geteilt, um zu ermöglichen, eine begrenzte Anzahl an Bildern herunterzuladen. Diese enthalten einen Counter `value` der threadsicher um 1 verringert wird. Wird 0 unterschritten, brechen die Tasks ab. Dabei muss auf den vollständigen Download gewartet werden, um sicherzugehen, dass die Bilder existieren.

Die Grundidee des Downloadvorganges entstammt dem [Pythonscript](#) von Open Images Dataset.

## 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>
- Pythonscript Open Images Dataset V6 <https://raw.githubusercontent.com/openimages/dataset/master/downloader.py>

## Anhang

### Screenshots des Trainings

```

R:\Transfer\Softwareentwicklung_Github\Finalabgabe\ConsoleApp\bin\Debug\net5.0\ConsoleApp.exe
Willkommen in der Konsolen-App zur Bildklassifizierung auf Grundlage von Machine Learning
Möchten Sie (1) Bilder kategorisieren oder (2) das Modell neu trainieren?
Wie viele Bilder sollen pro Kategorie heruntergeladen werden? (min. 1)
500

R:\Transfer\Softwareentwicklung_Github\Finalabgabe\ConsoleApp\bin\Debug\net5.0\ConsoleApp.exe
Willkommen in der Konsolen-App zur Bildklassifizierung auf Grundlage von Machine Learning
Möchten Sie (1) Bilder kategorisieren oder (2) das Modell neu trainieren?
Wie viele Bilder sollen pro Kategorie heruntergeladen werden? (min. 1)
500
Auswahl der Kategorien, bitte insgesamt mindestens zwei Auswählen!
Bitte eingeben, was in der Kategoriebezeichnung enthalten sein soll!
pizza
0: /m/0663v: Pizza
1: /m/068_x: Pizza cheese
2: /m/06wyp: California-style pizza
3: /m/08ks85: Pizza cutter
4: /m/09yjm8: Sicilian pizza
5: /m/0dlr0: Pizza stone
6: /m/0h8lq3g: Pizza pan
Bitte Kategorienummer eingeben oder -1, um Eingabe neuzustarten, bei mehreren mit Leerzeichen getrennt
0 3 6

R:\Transfer\Softwareentwicklung_Github\Finalabgabe\ConsoleApp\bin\Debug\net5.0\ConsoleApp.exe
10: /m/038czg: Jet bridge
11: /m/04229: Jet engine
12: /m/042rz: Jetsprint
13: /m/047cmpg: Oldsmobile jetstar 1
14: /m/04gh6cp: Ecojet concept car
15: /m/0638m: Coptera
16: /m/07_nj6: Hyundai trajet
17: /m/09fpwv: Daihatsu hijet
18: /m/09n5x: Trijet
19: /m/09n0k: Twinjet
20: /m/0b0dx: Audi roadjet
21: /m/0c3fma: Hudson jet
22: /m/0dd6m: Inkjet printing
23: /m/0fhqgn: Learjet 35
Bitte Kategorienummer eingeben oder -1, um Eingabe neuzustarten, bei mehreren mit Leerzeichen getrennt
0
Nach neuer Kategorie suchen [y/n]: n
Durchsuchen der ImageIDs.....
Dataset Pizza wird heruntergeladen nach: R:\Transfer\Softwareentwicklung_Github\Finalabgabe\tmp\Pizza
Download wird gestartet, Timeout beträgt 30 Sekunden
Bilder in Queue: 1930
Download wird gestartet, Timeout beträgt 30 Sekunden
Bilder in Queue: 1929
Download wird gestartet, Timeout beträgt 30 Sekunden
Bilder in Queue: 1928
Download wird gestartet, Timeout beträgt 30 Sekunden
Bilder in Queue: 1927
Download wird gestartet, Timeout beträgt 30 Sekunden
Bilder in Queue: 1926

R:\Transfer\Softwareentwicklung_Github\Finalabgabe\ConsoleApp\bin\Debug\net5.0\ConsoleApp.exe
Bild: 9732629a04bf9876.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: 9d672d91c8d1d240.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: a32008e6e6da2c06.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: a3568894bf36b1d5.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: a705df369564a056.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: b512ae0e70b5f568.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 98,6
Bild: c24ba67db0192a74.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: c4853d7d9171ed3d.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Bild: cb8e32ae6f64c4c.jpg Gelabelt Als: Jet aircraft Bestimmt Als: Jet aircraft Sicherheit: 100,0
Statistiken zum Training:
LogLoss: 0,28802178604179757
PerClassLogLoss: 0,16834191674144827 , 2,5487618197474187 , 2,3334995542473855 , 0,002692288712177612
Sie können das Modell jetzt speichern. Unter welchem Namen soll das Modell gespeichert werden? (Ohne Extension)
Modell
Das Modell ist unter R:\Transfer\Softwareentwicklung_Github\Finalabgabe\TensorFlow\Model1.model gespeichert
Modell erfolgreich trainiert!
Es wurden folgende Kategorien trainiert:
***Pizza***
***Pizza cutter***
***Pizza pan***
***Jet aircraft***
Möchten Sie das (oder andere) Modelle direkt zur Klassifizierung nutzen? [y/n]: y
Wählen Sie bitte unter den folgenden Modellen aus:
Folgende Modelle sind vorhanden:
Modell: "Ampel" mit Kategorien: "Cat-Traffic light- (Nr. 0)"
Modell: "Modell" mit Kategorien: "Pizza-Pizza cutter-Pizza pan-Jet aircraft- (Nr. 1)"
Bitte Entscheidung für ein Modell durch Eingabe der jeweiligen Nummer treffen und mit Enter bestätigen
Eingabe muss wiederholt werden, wenn inkorrekt
  
```

Abb. 4: Screenshots des Programmablaufes des Trainings



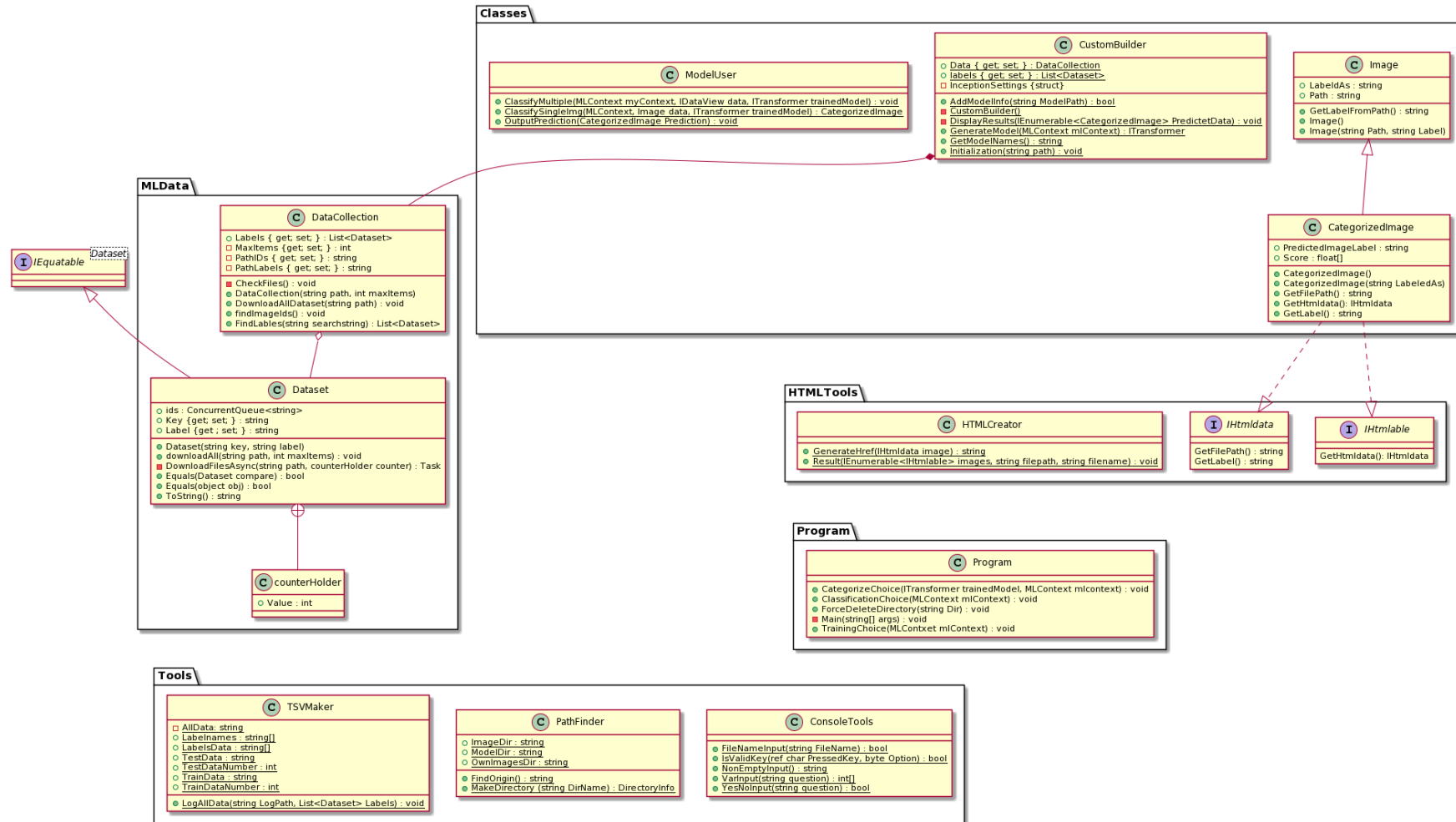


Abb. 5: Klassendiagramm des finalen Programms