|  |  |
| --- | --- |
|  | |
| Image Classification  Report | |
| **Robin Ramaekers, Britt De Keyser, Syan Delbart – 3AI01/2DI** | Big Data ITFactory |
|  |
| Academic Year 2022-2023  Campus Geel, Kleinhoefstraat 4, BE-2440 Geel | |



**Table of contents**

[Introduction 4](#_Toc121851415)

[1 Collecting / Selecting dataset 5](#_Toc121851416)

[1.1 Webscraper 5](#_Toc121851417)

[1.2 EDA on the data 5](#_Toc121851418)

[2 Modeling using FastAI 6](#_Toc121851419)

[2.1 The simple model 6](#_Toc121851420)

[2.2 Advanced model 6](#_Toc121851421)

[3 Deploy via Streamlit 7](#_Toc121851422)

[4 Extra: Deploying the model on flask 8](#_Toc121851423)

[Source List 9](#_Toc121851424)

[5 Attachments 10](#_Toc121851425)

[5.1 Webscraper 10](#_Toc121851426)

[5.2 Streamlit 13](#_Toc121851427)

[5.3 Jupyter notebook for the AI models 14](#_Toc121851428)

[5.4 Flask API code 14](#_Toc121851429)

# Introduction

The main goal of this assignment is an image classification app in Streamlit. We will achieve this by using FastAI.

We had to collect a dataset of images that can be categorized in at least 5 different classes. We needed to make sure these were challenging enough, to see if our model could also handle more challenging images. We ensured that the quality of our dataset was good, by performing an EDA.

After this we used FastAI modelling techniques to build a state-of-the-art deep learning model that is able to classify the images

Following, we made an interactive web application using Streamlit, where we can test a pre-saved model, by uploading a new image and have the application tell you the class the image belongs to. We also created an API endpoint around our model, by using Flask.

# Collecting / Selecting dataset

For this assignment, we had to collect a dataset of images that could be categorized in at least 5 different classes. We chose to base our assignment on different types of bikes such as mountain bike, gravel bike, Eroica bike, speed pedelec and road bike.

## Webscraper

To collect these images, we used a web scraper.

With our Google Images web scraper, we had some difficulties in the beginning. While scraping data, the script stopped running because there were ads located between the images. Because of this, we created a web scraper for DuckDuckGo, to see if scraping through another search engine solved our problem.

DuckDuckGo had more irrelevant pictures that we would have to filter out, so we still chose to continue working with the Google Images web scraper, as this seemed to give more accurate results. We were able to fix our problem by filtering the class of the images.

You can find the code for our web scraper in the attachments below.

## EDA on the data

We also had to conduct an EDA on our images because of the topic bikes. There were a lot of images that contained humans, bike parts and posters. We chose to filter the images manually.

Example of dirty data:  
 

Source: <https://cdn2.hubspot.net/hubfs/414695/43952042_10157894147748275_6634612350668242944_o.jpg>

# Modeling using FastAI

Because our data is now cleaned, we can start with making our AI models. We will be using Google Colab with fastAI and Dataloader.

## The simple model

For our simple AI-model we used a simple restnet50 model with no fine-tuning. With this we were able to achieve an accuracy of 69% which is fine since the bikes have a lot of similarities. Still, we thought we could improve the model further.

## Advanced model

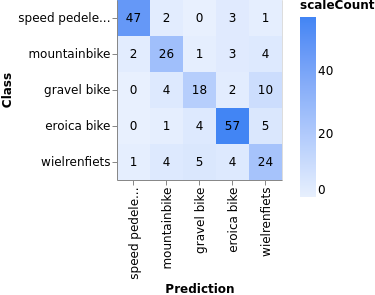
We decided to go further with the restnet50 model because it seemed like the best option for our classifications. We fined tuned the model so it used a batch size of 16, when this was done, we implemented 3 freeze epochs after this it would train 10 more epochs this resulted in an accuracy of 76% with minimal loss. Finally, we plotted the confusion matrix so we could compare our model to the Google teachable machine model.

Our model:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Google teachable machine:



As you can see we beat Google teachable machine!

You can find the link to our Colab notebook in the attachments

# Deploy via Streamlit

After the advanced model was trained, we exported and downloaded it so we could later use it in our Streamlit environment. Deploying the model to Streamlit was straightforward. The only issue we encountered was with our pickle file we couldn’t import it properly this was later fixed by importing pathlib. Our final Streamlit app looks like this: Afbeelding met tekst, schermafbeelding, monitor, scherm

Automatisch gegenereerde beschrijving

You will need to upload an image of a bike and press classify, our AI model will make the prediction and show its confidence.

You can find the code for Streamlit in the attachments below.

# Extra: Deploying the model on flask

For the extra we created an API endpoint around our model, by using Flask. We tested our API via Postman. You can find the code in the attachments. Afbeelding met fiets, grond, geparkeerd, buiten

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Source List

<https://medium.com/featurepreneur/deploy-a-machine-learning-model-using-streamlit-a7ef163c19b0>

<https://towardsdatascience.com/deploy-a-machine-learning-model-using-flask-da580f84e60c>

<https://towardsdatascience.com/how-to-build-a-machine-learning-api-using-flask-2fb345518801>

<https://stackoverflow.com/questions/68303551/typeerror-object-of-type-tensor-is-not-json-serializable-dict-to-json-error-i>

<https://www.vertopal.com/en/convert/ipynb-to-pdf>

<https://stackoverflow.com/questions/52244707/pip-unable-to-install-fastai>

# Attachments

## Webscraper

#imports

import bs4

import requests

from selenium import webdriver

import os

import time

from selenium.webdriver.common.by import By

#creating a directory to save images

folder\_name = 'images'

if not os.path.isdir(folder\_name):

    os.makedirs(folder\_name)

#function to download the images

def download\_image(url, folder\_name, num):

    # write image to file

    reponse = requests.get(url)

    if reponse.status\_code==200:

        #wil = williams you can change this per category

        with open(os.path.join(folder\_name,'ped'+ str(num)+".jpg"), 'wb') as file:

            file.write(reponse.content)

#path to your chromedriver

#make sure its is stored in a PATH variable in your enviromental settings

chromePath=r'C:\Users\britt\Documents\chromedriver.exe'

driver=webdriver.Chrome(chromePath)

#The search string

#edit this manually for different categories

#keywords after q=. seperrated by a + sign

search\_URL = "https://www.google.com/search?q=speed+pedelec&source=lnms&tbm=isch"

driver.get(search\_URL)

#code to get through the TOS

className = "VfPpkd-LgbsSe-OWXEXe-k8QpJ"

time.sleep(2)

print(className)

driver.find\_elements(By.CLASS\_NAME, className)[2].click()

#wait 5 seconds

time.sleep(5)

#scroll to the bottom of the page and enter a keyword

a = input("Waiting...")

#Scrolling all the way up

driver.execute\_script("window.scrollTo(0, 0);")

#Find the containers of the images

page\_html = driver.page\_source

pageSoup = bs4.BeautifulSoup(page\_html, 'html.parser')

containers = pageSoup.findAll('div', {'class':"isv-r PNCib MSM1fd BUooTd"} )

print(len(containers))

len\_containers = len(containers)

#Download the images that are in these containers

for i in range(1, len\_containers+1):

    #xPath of the image

    xPath = """//\*[@id="islrg"]/div[1]/div[%s]"""%(i)

    error = False

    try:

        previewImageXPath = """//\*[@id="islrg"]/div[1]/div[%s]/a[1]/div[1]/img"""%(i)

        previewImageElement = driver.find\_element(By.XPATH,previewImageXPath)

        #Get the image source

        previewImageURL = previewImageElement.get\_attribute("src")

    except:

        print("No image, is advert")

        error = True

        continue

    #click the image

    driver.find\_element(By.XPATH,xPath).click()

    timeStarted = time.time()

    while True:

        imageURL = ""

        try:

            imageElement = driver.find\_element(By.CLASS\_NAME,"n3VNCb")

            imageURL= imageElement.get\_attribute('src')

        except:

            print("Image not existing")

        if not imageURL or imageURL != previewImageURL:

            break

        else:

            currentTime = time.time()

            if currentTime - timeStarted > 10:

                print("Timeout! Will download a lower resolution image and move onto the next one")

                break

    #Downloading image

    try:

        if not error:

            download\_image(imageURL, folder\_name, i)

            print("Downloaded element %s out of %s total. URL: %s" % (i, len\_containers + 1, imageURL))

    except:

        print("Couldn't download an image %s, continuing downloading the next one"%(i))

## Streamlit

import streamlit as st  
import os  
import fastai  
  
import pathlib  
temp = pathlib.PosixPath  
pathlib.PosixPath = pathlib.WindowsPath  
  
from fastai.vision.all import \*  
from fastai.vision import \*  
from fastai.data.external import \*  
  
import pandas as pd  
from fastai.learner import load\_learner  
  
from zipfile import ZipFile  
  
  
class Predict:  
 def \_\_init\_\_(self, filename):  
 self.learn\_inference = load\_learner(Path() / filename)  
 self.img = self.get\_image\_from\_upload()  
 if self.img is not None:  
 self.display\_output()  
 self.get\_prediction()  
  
 @staticmethod  
 def get\_image\_from\_upload():  
 uploaded\_file = st.file\_uploader("Upload Files", type=['png', 'jpeg', 'jpg'])  
 if uploaded\_file is not None:  
 return PILImage.create((uploaded\_file))  
 return None  
  
 def display\_output(self):  
 st.image(self.img.to\_thumb(500, 500), caption='Uploaded Image')  
  
 def get\_prediction(self):  
  
 if st.button('Classify'):  
 pred, pred\_idx, probs = self.learn\_inference.predict(self.img)  
 st.write(f'Prediction: {pred}; Probability: {probs[pred\_idx]:.04f}')  
 else:  
 st.write(f'Click the button to classify')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 file\_name = 'Bikes.pkl'  
 st.title("Welcome to the bike classifier")  
  
 predictor = Predict(file\_name)

## Jupyter notebook for the AI models

Code:

<https://colab.research.google.com/drive/15xpfB2nOYYBVxMg8Bmg1SxZyHGeVENgG?usp=sharing>

## Flask API code

import io  
import flask  
import string  
import time  
import os  
import numpy as np  
import tensorflow as tf  
from PIL import Image  
from flask import Flask, jsonify, request  
import pathlib  
temp = pathlib.PosixPath  
pathlib.PosixPath = pathlib.WindowsPath  
import pandas as pd  
from fastai.learner import load\_learner  
  
from zipfile import ZipFile  
  
from fastai.vision.all import \*  
from fastai.vision import \*  
from fastai.data.external import \*  
  
model = load\_learner('Bikes.pkl')  
  
  
app = Flask(\_\_name\_\_)  
@app.route('/', methods=['GET'])  
def index():  
 return 'Machine Learning Inference'  
  
@app.route('/predict', methods=['POST'])  
def makepredict():  
 file = request.files.get('file')  
 image =file.save('./image.png')  
 prediction = model.predict('./image.png')  
  
  
 return jsonify('The prediction is ',prediction[0])  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 modelfile = 'Bikes.pkl'  
 model = load\_learner(modelfile)  
 app.run(debug=True, host='0.0.0.0')