LUMEN Data science

2022

Technical Documentation

1 Contents

1 Contents 2

2 Technical issues 2

2.1 The problem 2

2.2 Solution presentation 3

3 Design choices 3

3.1 Python 3

3.2 Django 4

3.3 ReactJS 4

4 REST API 4

5 Web application 6

5.1 Starting the web application 6

5.2 Overview 7

5.3 Functionality 7

6 Code organization 8

7 Sources 8

2 Technical issues

2.1 The problem

The main problem we were tasked with solving was guessing the geographic location (latitude and longitude coordinates) of four pictures taken from Google's Street view. Our solution is constructed in such a way so it can work with one or multiple pictures.

2.2 Solution presentation

To solve this problem, first we had to design an AI architecture that would perform well in the given task. The analysis and experiments performed in order to find the optimal model, which proved to be the EfficentNetB4 model, are presented in the Project documentation.

Secondly, we had to build a REST API which would enable users to interact with our model.

Additionally, we built a web application for the easier use of our solution model. The already built REST API worked as the back end of our site, while we built the front end using react JS.

In section 3 Design choices we present the technologies used in our execution of the described solution idea.

3 Design choices

3.1 Python

We used Python to read, process and visualize data, create and train our neural network and to handle the back-end for our web page. We decided to use Python for our machine learning needs because it offers good existing implementations of ML methods. It is also commonly used in data science and processing. Alongside vanilla Python, major libraries used are: NumPy, Matplotlib, OpenCv, TensorFlow and Keras.

NumPy, a Python library for working with large, multi-dimensional arrays and matrices, is used for most of our data processing in Python.

Matplotlib, a plotting library for Python and NumPy, was used to plot the original data onto a map allowing for a clear view of the data.

OpenCv, a library commonly used for computer vision, is used for loading and reshaping images. It is also used in the creation of the image showing the network's guess in the web interface.

TensorFlow, a Python library built for complex numeric calculations and offering excellent machine learning integration, and Keras, an open source API enabling fast and easy neural network implementations, are used for building and evaluating our EfficentNetB4 model.

We had settled on the EfficentNetB4 model as the best performing model following our experiments, which are detailedly presented in the Project documentation.

3.2 Django

Django is a Python-based open-source web framework that follows the model-template-view architectural pattern. While Django is mostly used for creating complex, database driven websites we used it here because most of our project was already written using Python, so it made sense to use it for the back end of our project as well.

3.3 ReactJS

React is an open-source JavaScript library for building user interfaces that uses a special syntax called JSX which allows mixing HTML with JavaScript. All components are loaded upon starting the page but are only rendered if needed, which makes reloading and rerouting pages unnecessary. This property makes React applications fast and simple which is why we chose React to create the front-end for our web application. Another reason why we used ReactJS is the fact that some team members had more prior experience with it, compared to other alternatives.

4 REST API

REST API is a way of accessing web services in a simple and flexible way without having any processing. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP requests. A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, image or JSON.

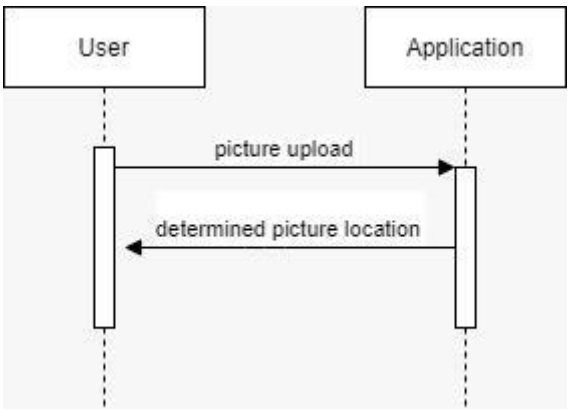

Image 1 - Sequence diagram

Our application consists of only one POST request. The user uploads 1 or more pictures via the HTTP request, which are then processed and the response with the geographical coordinates is sent. A visual representation of that can be seen in the sequence diagram in *Image 1*.

5 Web application

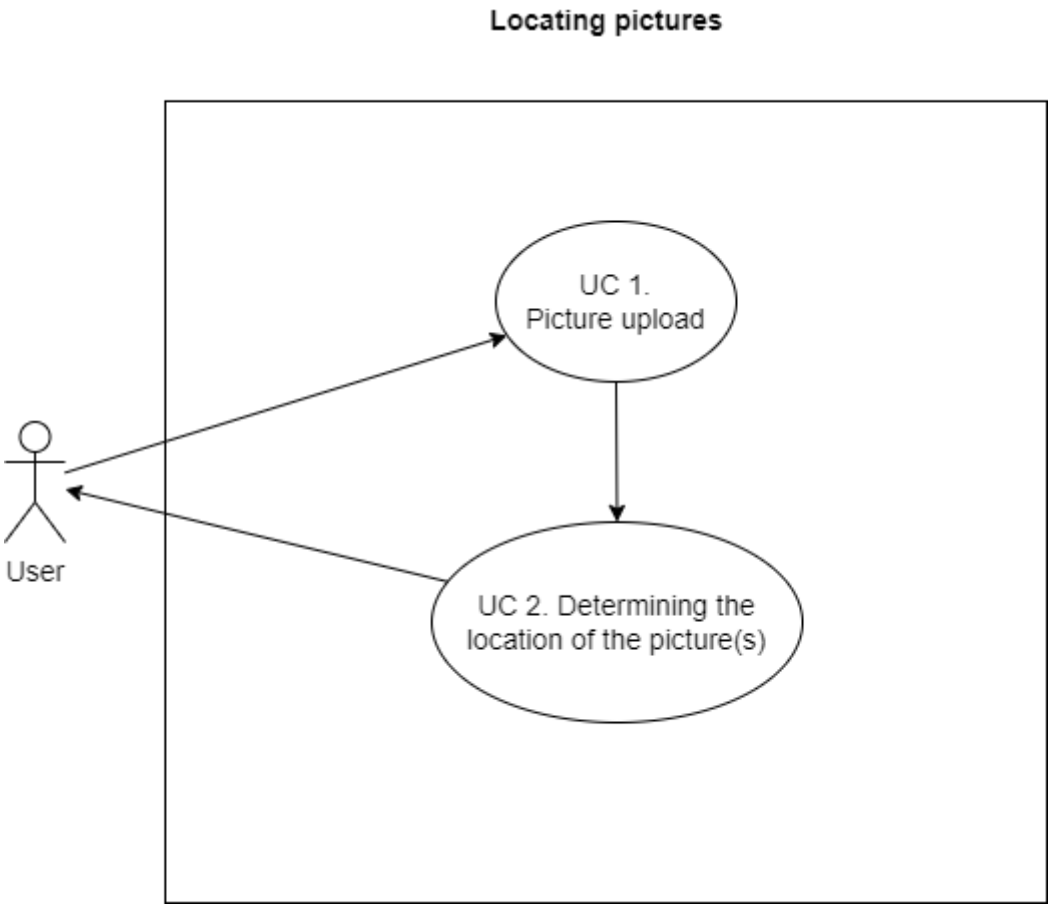

Image 2 - Use case diagram

Image 2 represents the actions of the user. The user has only one action and that is to upload pictures. This triggers the second action, determining the location, which returns the response back to the user,

5.1 Starting the web application

To start the web application, it is first necessary to configure the virtual environment (https://docs.python.org/3/library/venv.html) and install Django (https://www.stanleyulili.com/django/how-to-install-django-on-windows/) and the required libraries. Run the following commands in the backend root folder:

- *"pip install django"*
- *"pip install django-extensions"*
- *"pip install opencv-python"*
- *"pip install tensorflow"*
- *"python -m pip install django-cors-headers"*

To start the application's backend, the following two commands should be run in the "backend" folder: "python manage.py migrate" and "python manage.py runserver", as well as the following two commands in the "frontend" folder: "npm install" and "npm run".

5.2 Overview

A loaded page should be displayed like the one shown in *Image 3*. The main elements are the "Choose" and "Upload" buttons.

"Choose" button opens up a file explorer window, where you can choose the picture files (one or more pictures can be submitted) based on which a prediction of a location depicted in them will be made.

"Upload" button sends the chosen picture files to the ML model in order to make a prediction. It is important to note that this step may take some time to complete, especially if the computer the code is being run on does not have a GPU.

Image 3 - web application before a prediction

5.3 Functionality

Once the "Upload" button is pressed, it sends a POST request from frontend of the web application to the backend, where views.py accepts the request and calls main.predict_coords. The function's output, a JsonResponse is returned to the frontend containing the geographic coordinates predicted by the model. The model's guess is displayed and visualized as shown in Image 4.

Image 4 - web application after a prediction

6 Code organization

The web application code is split into two parts - the back end and the front end. The back end of the project (REST API and other functionalities) is located in the “backend” folder, while the code dealing with the front end is located in the “frontend” folder. The model, as well as the functions for predicting the coordinates are also located in the “backend” folder.

7 Sources

Image 1 - produced by team members

Image 2 - produced by team members

Image 3 - produced by team members

Image 4 - produced by team members