Tampere University of Applied Sciences

Retrievimage - Online Image Retriever

Tuan Nguyen

BACHELOR'S THESIS
May 2023

Bachelor of Engineering
Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Software Engineering

TUAN NGUYEN:
Retrievimage - Online Image Retriever

Bachelor's thesis 38 pages, appendices 5 pages
May 2023

The purpose of this project was to provide a solid foundation of knowledge on image retrieval systems, including necessary theoretical definitions, components of such applications, and basic concepts of image retrieval. More specifically, it aimed to create a schema, some instructions, and a starting point to quickly pick up for prototyping machine learning models. From that, the idea of an online platform that was able to match the target with its equivalent in the set, given a collection of images and a query image, was decided.

To achieve this, Python 3 was chosen as the main programming. The application was structured according to a pipeline that was particularly designed to integrate machine learning models into web applications, and further on, continuous delivery for machine learning. However, instead of training a model from end-to-end, Transfer Learning was introduced, and the idea of using a pre-trained model was considered better and saved time and resources. The pre-trained model put into use was from OpenVINO, an open-source toolkit boosting the performance of machine learning models significantly. The program was meant to be set up as an online service running on a server that could handle uploads of images from users, and exchange information with them via API endpoints. Every time the app was shut down, it automatically erased all the stored data, so there was no need to implement authentication functionality at the moment. The server was developed in Python with Flask as the web framework, and the Jinja2 template engine for rendering the user interface.

In the end, the web application was able to achieve the initial goals set for it. At the bare minimum, the expected counterpart of the target image was successfully found by the image retrieval model, with satisfactory accuracy, then presented on the website's front end. Literature in the field, journals, books, articles, and research papers were studied and referenced for conceptual sections of this thesis. On the other hand, on some occasions, the model failed to capture the expected result, as images were not uniformly pre-processed before being fed into the model. It was plain to see that larger image batches required more predicting and rendering time. Examples were described to demonstrate the work in the Appendix section. From this, remarks were given, and enhancements were suggested to improve the overall production of this online image retriever.

Key words: machine learning, openvino, web development, image retrieval, python, flask

CONTENTS

ABBREVIATIONS AND TERMS

API	Application Programming Interface
arXiv	A free, online repository of academic papers, often preprints, in multiple fields of studies
CBIR	Content-based Image Retrieval
CLI	Command line interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSS	Cascading Style Sheets
FM 23	Football Manager 2023
GET	A method of REST API to obtain necessary information legally from a service
Git	Free and open-source distributed version control system authored by Linus Torvalds for software development
GitHub	online platform providing hosting for software development and version control using Git
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
JPEG	Joint Photographic Experts Group
MobileNetV2	A convolutional neural network architecture designed by Google for efficient mobile and embedded vision applications
OpenVINO	Open Visual Inference and Neural network Optimization
PNG	Portable Network Graphics
POST	A method of REST API to write or register information legally to a service
REST	Representational State Transfer
REST API	API conforming to the contraints of REST architectural styles
RGB	Red-Green-Blue color model
TAMK	Tampere University of Applied Sciences

UI	User interface
VPU	Vision Processing Unit

1 INTRODUCTION

Over the last few years, the fields of machine learning and image retrieval have experienced remarkable growth, offering creative and pragmatic solutions to intricate challenges. With the emergence of powerful learning algorithms and the rise of extensive data, machine learning has emerged as a significant driving force behind countless technological breakthroughs, extending from speech-based assistants to self-driving automobiles (Goodfellow, Bengio & Courville, 2016). Meanwhile, image retrieval approaches have empowered users to shift through and inspect vast image databases, thereby facilitating novel applications, including facial recognition, object detection, and content-based image retrieval (Smeulders, Worring, Santini, Gupta & Jain, 2000).

As a consequence of these accomplishments, there has been a marked enhancement in the construction of internet-enabled applications that apply machine learning and image retrieval. Among these, web-based image retrieval systems are a typical representation, which allow users to navigate and access images from a vast database using distinct search criteria such as color, texture, and shape (Baeldung, no date). The application can be employed in diverse domains, spanning from commercial transactions and social media to medical services and security.

This thesis aimed to develop a web-based image retrieval application leveraging the Python-Flask duo and image retrieval methodologies. More specifically, the application capitalizes on deep learning models such as Convolutional Neural Networks (CNNs) to extract attributes from images and conduct similarity assessments, enabling the retrieval of comparable images from a massive database. The objective of this project was to create a web-based platform that allows users to upload images and conduct image-to-image searches within a database, catering to those looking for close or identical images. Through this thesis, we aim to make an impact in the thriving field of image retrieval, while illustrating the practical applications of machine learning in real-world scenarios.

2 BACKGROUND AND OBJECTIVE

2.1 Project Background

Image retrieval is a term describing the process of acquiring an image in the database. There has been research and comparisons on how to return the most accurate result given the conditions of the input query. Up to now, the most widely used and effective method for image retrieval demonstrations is content-based image retrieval or CBIR. How CBIR works is that model sorts out the desired image stored in a database that is most similar to the query image based on shapes, colors, texture, and spatial information. (Baeldung, no date)

Over recent years, machine learning has been the talk of the town from all over the globe, and research in the field has been hotter than ever. In 2021, an average of about 15136 papers were submitted to arXiv per month, accounting for 33.6%, which was the highest among all categories (arXiv info, no date). And in 2022, according to ame5, a contributor on arXiv blog, the page reached a milestone of more than 2 million articles in total, and up to 1200 new submissions every day (ame5, 2022). This has led to inventions and discoveries of never-seen-before aspects of machine learning like semi-supervised learning, self-supervised learning, and so on, concentrating more on the architecture of models able to perform tasks of both computer vision and natural language processing.

However, in 2018, there was a post on University World News mentioning the reason behind the explosion of research publications and the consequences it brought about. According to Philip G Altbach and Hans de Wit (2018), academic publishing experienced a crisis because of the pressure on top journals, emergence of predatory publishers, and focus on unnecessary publications. The growing trend of requiring doctoral candidates to publish their research in academic journals and replace traditional dissertations made a huge contribution to the explosion of research publications, which led to difficulties in evaluation and peer reviews.

With a total of 181630 research papers being published on arXiv platform during the year 2021 (arXiv info, no date), from the perspective of Philip G Altbach and Hans de Wit, it is obvious that only a fraction of publications get the most attention and applied to real-life applications. The common environment settings used for research are usually ideal and equipped with some of the most optimal tools and hardware for computing and processing, which makes it almost impossible to replicate when put into use with ordinary and affordable setups, and leave most papers unnoticed with little to no impact.

2.2 Project Goal

For a long time, individuals have employed text-based queries to search for images. However, this method is not as effective as using images to search for images. This is because it depends on the user's ability to accurately describe the image in words, which can be challenging, especially if the user is unfamiliar with the technical vocabulary used to describe images. Moreover, searching for images using text necessitates that the image has been properly annotated with appropriate keywords and metadata. However, by shifting to content-based image retrieval, the limitations of text-based queries can be overcome, as this technique relies on the image's visual features to search for similar images rather than relying on keywords or metadata (Baeldung, no date).

Nowadays, websites are a no-brainer for us when it comes to displaying, accessing, and gathering information, latest news all around the world. Thanks to the advancements of web development alongside machine learning frameworks over the recent years, with just some steps, people that are not familiar with machine learning can still get a taste of the model without in-depth knowledge of software engineering through a web-based user interface.

With that being said, this project is conducted to provide users with an environment to experiment with how image retrieval looks in the form of a web application. It is achieved by applying OpenVINO's Image Retrieval model, which is backboned by one of Google's most famous architectures for mobile models, MobileNetV2 (Intel, no date).

2.3 Technology Stack

2.3.1 Python

Python was first introduced to the world by Guido Van Rossum in 1991. It is a high-level interpreted programming language, and arguably the world's most popular programming language because it is beginner-friendly and pragmatic language. The language can be used to develop server-side applications with frameworks like Django, desktop graphical user interfaces, or scientific and numeric analysis. (Fireship, 2021)

Ever since the booming of Big Data, Artificial Intelligence, and Machine Learning has always been the top news topics when it comes to the field of Computer Science. With its ease to learn, use, and understand as well as its numerical and analytical power, Python takes first place as the programming of choice implementations of algorithms, models, or libraries related to machine learning and data visualization in both research and practice.

Besides the libraries and frameworks available for machine learning, there is plenty of tools and support available for learning and using Python because it also has a sizable and vibrant developer community. (Python, no date)

2.3.2 Flask

Flask is a Python-based micro-framework designed for web application development. Its micro-framework feature, which results in lightweight and flexible characteristics, means that it does not require any specific frameworks or tools. As per the official Flask documentation (2020), "Flask is user-friendly and an excellent tool for building web applications and sites." By prioritizing simplicity and flexibility, Flask offers developers a faster path to creating web applications.

Flask can be used to create web applications that let users interact graphically with machine learning models. Besides, Flask operates on the server rather than in web browsers of users due to its server-side characteristic. When using machine learning models, this is an advantage because it enables the model to be trained and operated on a server, which may have more resources than end devices. (Flask, no date). Although Javascript, which is mainly a client-side language, can be used for machine learning purposes, it is not as effective as Python due to a large number of dynamic libraries and frameworks specifically designed for creating machine learning models, so it may be more convenient to use the Flask-Python duo for web applications involving machine learning models (Grinberg, 2018).

2.3.3 Jinja

Jinja, or jinja2, is a powerful and widely used templating engine for Python that is designed to be fast, flexible, and easy to take up (Jinja, no date). It is specifically the number one contender when used alongside the Flask web framework (Grinberg, 2018).

How Jinja works simply is the template contains so-called "special placeholders", and these enable developers to write code that has almost the same syntax as the Python programming language. After that, data is collected and passed to the template to render the final document. One of the key features of Jinja is the ability to support template inheritance, which allows easy reusing of common elements across multiple templates. Cutting down on the amount of repeated code can significantly ease the process of creating and maintaining a web application. Jinja is a useful tool for running dynamic web applications as it supports a broad variety of expressions, including mathematical and logical operators. (Jinja, no date)

2.3.4 OpenVINO

OpenVINO, or Open Visual Inference and Neural network Optimization, is an open-source toolkit developed by Intel that allows developers to optimize deep learning models that have already been pre-trained for plenty of hardware platforms, such as CPUs, GPUs, and VPUs. It, as well, supports the deployment of machine learning models to edge devices with efficient performance, low power consumption, and a compact footprint. Moreover, OpenVINO offers various pre-trained models that may be picked up to quickly get started with developing deep learning applications, without the need to implement them from scratch. (Intel, no date)

OpenVINO is extensively put into use in numerous applications regarding image and object recognition, facial recognition, and object detection. Developers can test and evaluate the performance of models on various hardware platforms using the demo suite included in the toolkit (Condon, 2018). Along with support for many popular frameworks like TensorFlow or PyTorch, OpenVINO toolkit also facilitates several libraries and APIs that have the capability to enhance the performance of deep learning models (Intel, no date).

3 IMAGE RETRIEVAL WEB APPLICATION REQUIREMENTS

3.1 Web Application Development

The fascinating history of web development can be traced back to the 1980s, when a British scientist named Tim Berners-Lee, who was working at CERN at the time, launched World Wide Web (WWW) in 1989, and the first web page and web server in 1991, which users can now still visit at info.cern.ch (CERN, no date). The history of web development is also intertwined with the development of the Internet. The 1990s saw the rise of web browsers like Netscape Navigator and Internet Explorer, which gave users access to websites and interact with them. Early websites were virtually built on HyperText Markup Language (HTML) and Cascading Style Sheets (CSS). Then came the introduction of Javascript to the world in 1995 which enabled developers to extend the limits of HTML and start creating dynamic web pages (MacManus, 2020). In recent years, the drastic emergence of mobile devices and mobile web has further transformed web development, making responsive designs increasingly essential, and allowing websites to adjust their layouts to fit different screen resolutions (Marcotte, 2011). Nowadays, thanks to their implementations and easy-to-get-started characteristic, web frameworks like React and Vue are preferred to build and scale web applications rapidly.

The growth and demand for web development have also increased significantly over the last several years. As per the US Bureau of Labor Statistics (no date), it is estimated that the employment of web developers tends to go up 23 percent during the period between 2021 and 2031, which is much quicker than the usual rate for all occupations. The remarkable explosion of digital marketing and e-commerce, as well as the widespread adoption of mobile devices, are driving this demand. The change towards remote work and online services has also been accelerated by the COVID-19 pandemic, increasing the demand for qualified web developers. Additionally, the growth of emerging technologies is opening new opportunities for web developers to innovate and produce cutting-edge web apps.

3.2 Functional Requirements

The functional requirements that are compulsory to function the web application in this project are as follows:

- Application must start when accessing the URL generated by either using Python and Flask from the command line or free public hosting services after deployment.
- Web app must show its initial state when users first access the link.
- Buttons in the application must be present sufficiently and suitably.
- Users must be able to choose any number of images to upload, upload them to the web app, and view the images under the form of a table with uploaded dates.
- Actions in each item on the list must work properly, meaning that each image should be able to be deleted or selected as target.
- Users should be able to add more images to the list by uploading new images.
- Radio buttons below the query image table should assist users in choosing the right number of images to show in the result section.
- "Predict" button must trigger the Image Retrieval model at the back and returns results to the frontend, if and only if a target image is decided.
- Each predicted image should have an according score calculated by distance from it to target image in the prediction table.
- "Refresh" button should be able to clear the current state, taking users back to the application's initial state.

3.3 Non-functional Requirements

The non-functional requirements that are compulsory to function the web application in this project are as follows:

- Application should have a table to display a list of query images that acts as a mock image database.

- Each item on the query image list should have information on its uploaded date, and actions to be deleted or selected as target.
- Target image and predicted results should appear separately as a prediction section.
- Application should have a table to display a list of predicted results taken from the query image list.

4 TECHNICAL IMPLEMENTATION

4.1 Planning

The first step in the technological execution of Retrievimage is to plan what can be created so as to demonstrate machine learning in general, and image retrieval to be specific, in production and how the product can achieve the desired goal. For this project to be successful, the following questions should be taken into account and analyzed:

- How should the image retrieval model be trained? Should we start from scratch, or pick up a pre-trained model on the internet?
- If we choose to start from scratch, how many resources do we need? How is the pipeline constructed? How is success defined and how long does it take for the model to reach expected results?
- What technologies should be picked up for developing the web application?
- Where should the images be temporarily stored when users upload them onto the web application?

After careful consideration, it is decided that Retrievimage will be constructed as a middleman to help find similar images given an input photo from clients. Using a website is excellent to demonstrate image retrieval in an interactive environment to those interested in playing around with applications of image retrieval and seeing the model function in production. Moreover, Retrievimage provides everyone with a template as a starting point to quickly pick up and start building an interface prototyping the machine learning model of choice.

4.2 Transfer learning

Transfer learning refers to a machine learning technique whereby a model pre-trained on a source task is subsequently utilized as a starting point to facilitate the learning of a target task (Pan & Yang, 2010). The pre-trained model serves as a starting point for the new task, with its learned internal representation being a feature extractor (Yosinski, Clune, Bengio & Lipson, 2014). This approach can significantly reduce the computational cost of training a new model from scratch while improving its accuracy and generalization ability. Transfer learning leverages the pre-existing knowledge learned by models trained on large and complex datasets, allowing for the recognition of generalizable features that can be applied to other related tasks (Weiss, Khoshgoftaar & Wang, 2016). Thanks to the efficiency and flexibility of pretrained models, the potential applications of transfer learning span across various areas, such as natural language processing, computer vision, and speech recognition, and the technique can significantly reduce the amount of data required for training new models, making it a valuable tool for addressing data scarcity issues (Oquab, Bottou, Laptev & Sivic, 2014).

4.3 Application Model

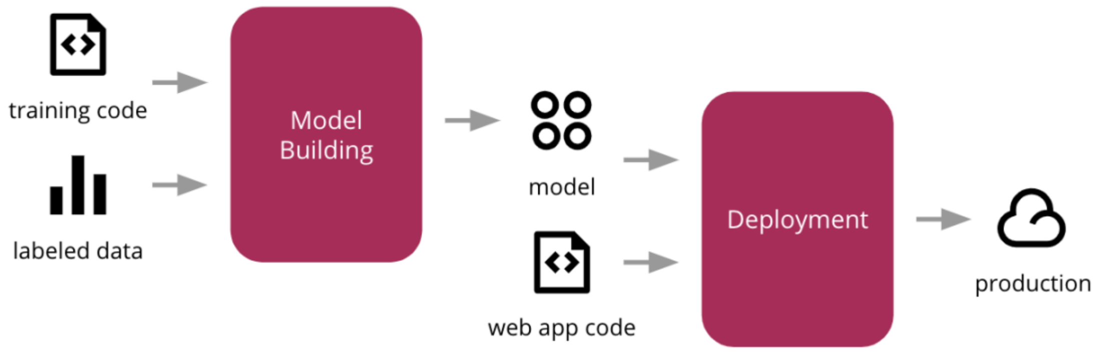


FIGURE 1. Pipeline to build and integrate an ML model into web application for production (Sato, Wider & Windheuser, 2019).

FIGURE 1 depicts the most common pipeline for a machine learning model from training to web development codebase integration, and deployment onto the Internet for public use. The pipeline begins with collecting data, preprocessing data using different techniques like data labeling and data cleaning, feature engineering, and writing the code to train the model from scratch. The preprocessed data is split into training, validation, and test sets to evaluate how model performs after the training process completes with the appropriate choice among various architectures and techniques. The trained model is then evaluated on the validation and test sets, and optimized with the help of hyperparameter tuning and other techniques. The web app code will be modified, and the machine learning model will be embedded into the web application for a specific task once optimized and ready to serve. The whole application is then packaged and deployed in a production environment for the purpose of generating predictions in real-time. This description is a shorter version big-picture, extended view of continuous delivery for machine learning, where production environment is continually monitored, model is incrementally improved by re-training with new online input data, feedback loops from user feedback, production metrics, and model performance metrics to keep the machine learning application from being ineffective and irrelevant, but all of them are beyond the scope of Retrievimage.

The application design of Retrievimage is modeled after the pipeline demonstrated above. However, the approach taken is different from the conventional way of training machine learning models from scratch. Instead, with the idea of Transfer Learning, a pre-trained model is used to attain the desired results more efficiently in terms of both time and resources. The rationale behind this decision is that creating a fully-functional and high-performing machine learning model requires a significant amount of time and effort to go through the end-to-end training, testing, and evaluation processes, which are conducted in multiple loops to allow the model to enhance its learning after each iteration, and can be both computationally expensive and time-consuming.

That being said, the concentration is primarily on developing Python code for machine learning model application and web development. The image retrieval model is referenced from the source code of a demo on GitHub, where the model helps to identify similar images in each frame of a video using dependencies and helper functions from OpenVINO™ Toolkit. The source code contains classes that build up the structure for an ImageRetrieval object, providing each instance with essential public methods to complete tasks such as handling input video or image lists, searching for images in the gallery, and processing images by computing embeddings and distances for each item. With this foundation in place, another Python file is created to serve the primary purpose of image retrieval, with a function defined to instantiate the ImageRetrieval class. The instance is then populated with appropriate arguments, mainly from user inputs and uploads, when they interact with the user interface. The program proceeds to process all the images, converting them into scalars or vectors, and returns the number of results based on the conditions specified by users.

Regarding the web development aspect, Flask-Jinja is a popular combination for Python developers because it is easy to use, fast, and straightforward for demo applications and prototypes. When the application is accessed for the first time, it automatically directs users to the home path, which is "/", and activates the home() method to create the initial user interface. Users can start interacting with the website by selecting images and uploading them to the site from this UI. Since the home route accepts both GET and POST methods and uploads are classified as POST requests, the home() method is activated again to process the inputs and save them in the app as a Python dictionary with items that are numbered and indexed at 0, each containing a file-name and upload date. Once the POST request has been properly processed, the method returns the processed variables to the HTML template for rendering.

One useful feature of Jinja in the frontend is its ability to incorporate inline Python code into HTML templates using curly brackets. As a result, it is much easier to list images on the user interface with just a few steps of Python for-loop. Each item on the list has specific actions that users can activate, and each of these actions directs users to its API endpoint and methods, which are delete() to remove an image, select() to designate an image as the target, and deselect() to remove the image from being the target. After images are uploaded, and a target is chosen, it is time for predictions. The predict() method manages everything related to image retrieval. When the "Predict" button is clicked, predict() is called, and in that function, it combines all the necessary values for the model and triggers the Retriever function with that set of arguments. Finally, the "/refresh" endpoint points to the refresh() method, which erases everything with one click of "Refresh", including the result section, and takes users back to the starting view.

5 CODE EXPLANATION

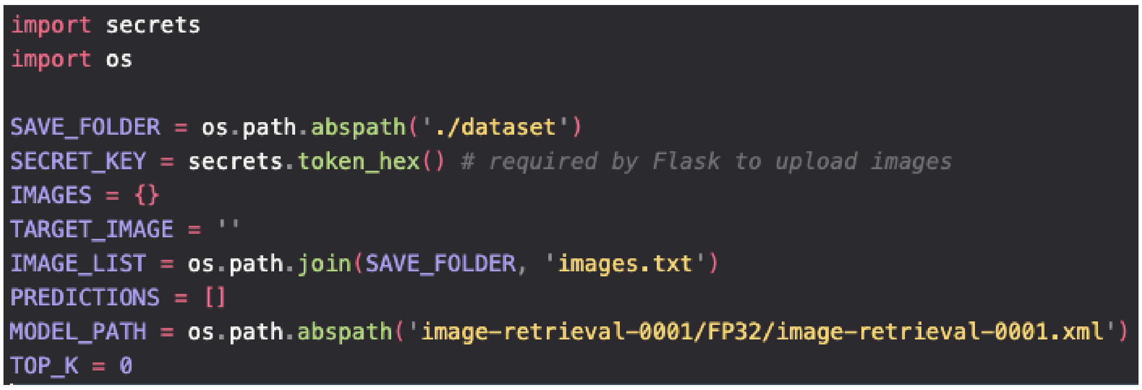
5.1 Introduction to the Image Retrieval Model of OpenVINO

The image retrieval model utilized in the web application runs on the backend and its code is derived from the official Image Retrieval Python demo implemented by OpenVINO. This demo provides users with step-by-step instructions on how to achieve the same level of performance and results using a CLI program provided by the user interface. In addition, there is also further information on the model's implementation from OpenVINO with maximum performance, color model, and input/output dimensions. (Intel, no date)

On the other hand, the demo was designed to handle target files in video format, which is not in line with the goal of this research paper. Therefore, adjustments have been made to enable processing of images as target files. Additionally, the user interface component of the program has been eliminated, so that when activated from the web user interface, the program quietly executes the complete process in the backend, returns the outcomes, and exhibits them on the frontend of the web application.

5.2 Configurations for Flask Web Application

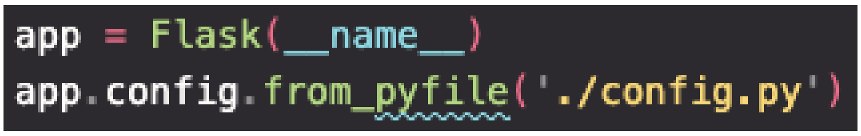
Environment variables are defined as a collection of parameters facilitating the operation of applications developed in Flask. Essentially, an environment is a directory or a placeholder that consists of all the prerequisites for an application to run successfully. Thanks to environment variables, users can modify built-in or even add more personalized settings and test the overall environment without affecting system-wide variables, as well as categorizing the type of environment for the Flask application directly from the codebase. (EDUCBA, no date)



```python
import secrets
import os

SAVE_FOLDER = os.path.abspath('./dataset')
SECRET_KEY = secrets.token_hex() # required by Flask to upload images
IMAGES = {}
TARGET_IMAGE = ''
IMAGE_LIST = os.path.join(SAVE_FOLDER, 'images.txt')
PREDICTIONS = []
MODEL_PATH = os.path.abspath('image-retrieval-0001/FP32/image-retrieval-0001.xml')
TOP_K = 0
```

FIGURE 2: Defining configuration variables.



```python
app = Flask(__name__)
app.config.from_pyfile('./config.py')
```

FIGURE 3: Importing configuration variables to main program.

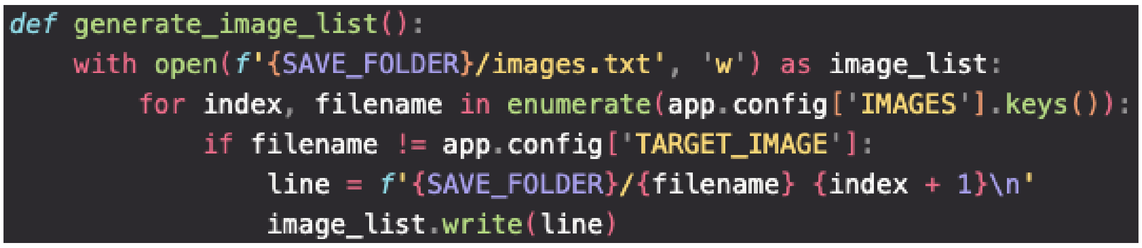
According to Todd Birchard of Hackers and Slackers, though considered to be fast and convenient, inline configuration of environment variables for Flask applications is never a good practice at all. In fact, this has the possibility to serious problems when configurations are mutated while the app is running, or leaks of sensitive values such as secret keys declared in the middle of source code. Another slightly better method is to update the variables in bulk, but this resembles the above-mentioned pattern only with a different look. (Birchard, 2018)

Then there is the method that I am using to configure environment variables. It is said to be the simplest way, yet clean and efficient, as it allows us to remove the mess from the main program by seperating configuration into its file, while maintaining the app logic flow. (Birchard, 2018)

5.3 Helper functions for Web Application

These modules play a big role during the coding phase of the application as they are defined at the top and enable reusing the functionality without the need to rewrite the whole procedure each time it needs referring to, which saves a whole lot of time at both ends.

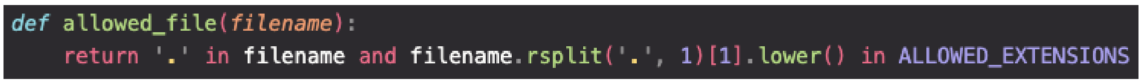
The condition for a procedure to be modularized is how frequently the function is called and how complex the procedure code can be. It is clear that a procedure should be modularized if it is used in many places for the same purpose, with just different inputs. There might be times when the procedure is only called once, but the logic is complicated, and it takes up a lot of spaces between the lines of code in the main program. That is a sign of modularization of the function for later references.



```python
def generate_image_list():
    with open(f'{SAVE_FOLDER}/images.txt', 'w') as image_list:
        for index, filename in enumerate(app.config['IMAGES'].keys()):
            if filename != app.config['TARGET_IMAGE']:
                line = f'{SAVE_FOLDER}/{filename} {index + 1}\n'
                image_list.write(line)
```

FIGURE 4: Helper function generating list of images.

The figure above demonstrates the process of getting the filenames from the dictionary of images and listing them into a text file, each associated with an index number, in such a way that this text file will later be sent to the backend, where the Image Retrieval model is triggered and processes the images mentioned in the text file.



```python
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

FIGURE 5: Helper function validating an uploaded image file.

Every time users upload files onto the website, procedure described in FIGURE 5 has the responsibility to check whether the files are valid images or not. It is done by simply separating the filename into two parts: name (before the dot), and extension (after the dot). Since a list of extensions is already created, the file extension is the only part that is put to the test. Additionally, both parts of the filename must be all converted to lowercase characters, which makes the checking process easier. Since we only process simple RGB images, the application only accepts common image extensions like PNG or JPEG.

5.4 Running Image Retrieval In CLI

From OpenVINO's demo for Image Retrieval in Python, the code has been referenced and modified in order to execute on an input image and display the results in form of log messages instead of processing frame to frame from a video or webcam. Also in Python, users can run the program in the command line interface given this simple syntax template:

"python retriever.py <path-to-target-image> <path-to-image-list> <k>"

- *path-to-target-image*: relative path to the target image file
- *path-to-image-list*: relative path to the text file containing image paths
- *k*: number of elements chosen as final result display

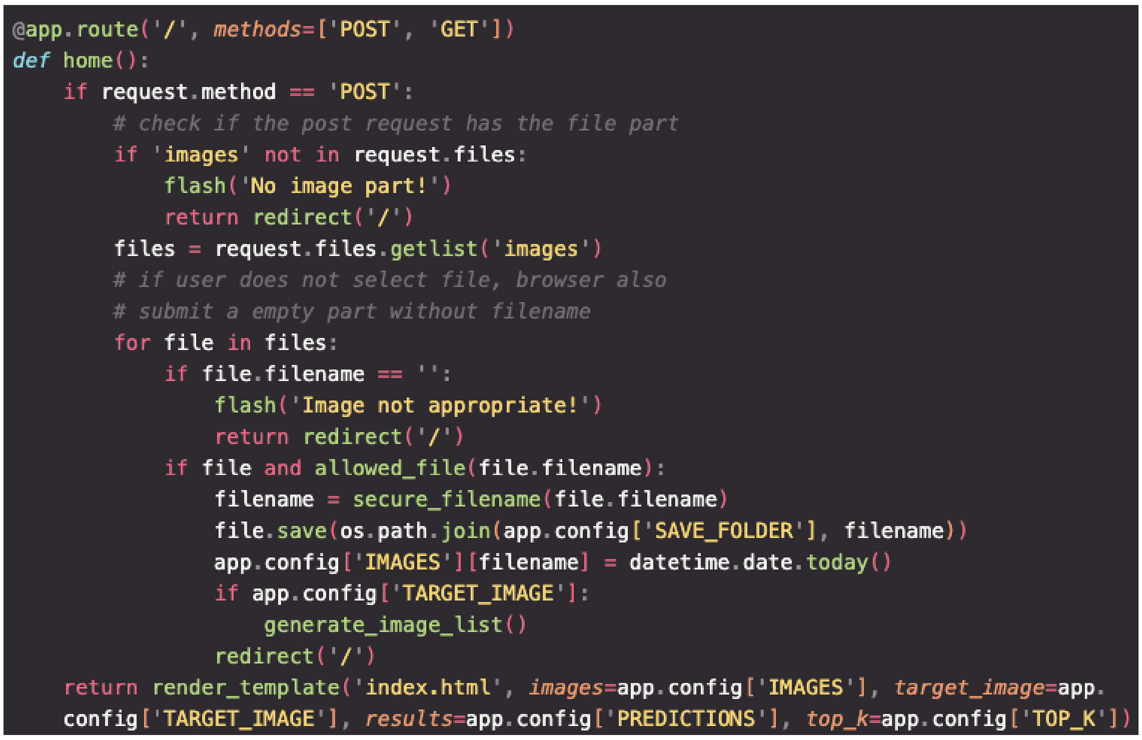
In this way, users have the flexibility to define the arguments of the Image Retrieval model without having it hard-coded into the program's codebase. At the end of the execution, there are prompts for the elapsed time - how long it takes for the whole procedure and result. Each element in the first component of result consists of the image filename associated with a metric score, defined by the distance from this image to the target image. In Python, list slicing still works perfectly even if the upper bound is set beyond the length. Therefore, unless *k* is set to 0 or image list is empty, application will work properly, and result list is prompted in ascending order of distance score.

5.5 Web Application Endpoints

5.5.1 Uploading Image List

When receiving a POST request from a user, the web application investigates the request body and searches for list of images (StackOverflow, 26.6.2019). Then, the list gets iterated through, and each element is examined from the filename to its extension (Flask, no date). If satisfied, every image is associated with the period of time it is uploaded to the app, so that the app can provide more information to the user interface. In Python, dictionaries are known for fast indexing of keys and values, equivalent to hashmaps, so all the image-date pairs are appended to a dictionary predefined at the top of the program.

This endpoint, at the same time, also serves as the main path, or homepage, for the application. In case of a GET request, such as a page refresh or a first-time visit to the page, *render_template* renders the HTML template along with some of the keyword variables given as arguments of the method (Flask, no date).

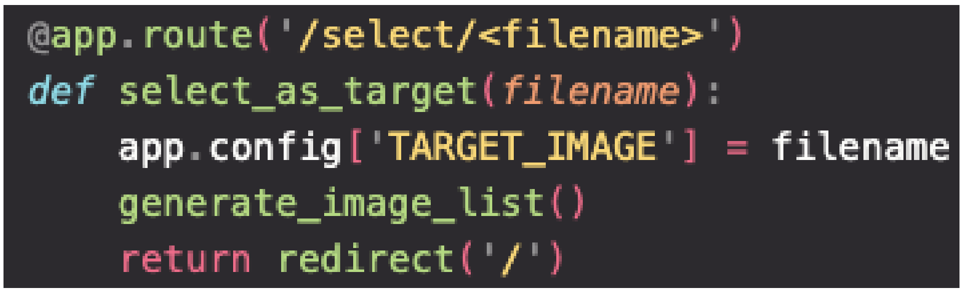


```python
@app.route('/', methods=['POST', 'GET'])
def home():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'images' not in request.files:
            flash('No image part!')
            return redirect('/')
        files = request.files.getlist('images')
        # if user does not select file, browser also
        # submit a empty part without filename
        for file in files:
            if file.filename == '':
                flash('Image not appropriate!')
                return redirect('/')
            if file and allowed_file(file.filename):
                filename = secure_filename(file.filename)
                file.save(os.path.join(app.config['SAVE_FOLDER'], filename))
                app.config['IMAGES'][filename] = datetime.date.today()
                if app.config['TARGET_IMAGE']:
                    generate_image_list()
                redirect('/')
    return render_template('index.html', images=app.config['IMAGES'], target_image=app.
    config['TARGET_IMAGE'], results=app.config['PREDICTIONS'], top_k=app.config['TOP_K'])
```

FIGURE 6: Image uploading and homepage rendering.

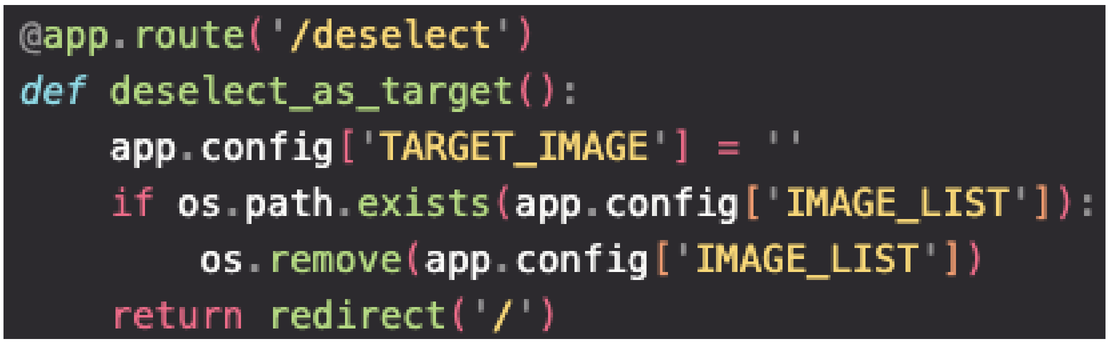
5.5.2 Choosing Target Image

Instead of implementing another file uploading field for the target image like how it is done with uploading the initial image list, it seems to be more convenient to give users the choice to select the target image right from the uploaded list. Generally speaking, there is no method clearly better or more efficient than the other, so it really comes down to the preference of the developers when choosing the suitable implementation for the project.



```
@app.route('/select/<filename>')
def select_as_target(filename):
    app.config['TARGET_IMAGE'] = filename
    generate_image_list()
    return redirect('/')
```

FIGURE 7: Selecting target image endpoint.



```
@app.route('/deselect')
def deselect_as_target():
    app.config['TARGET_IMAGE'] = ''
    if os.path.exists(app.config['IMAGE_LIST']):
        os.remove(app.config['IMAGE_LIST'])
    return redirect('/')
```

FIGURE 8: Deselecting target image endpoint.

The figures above are the behind-the-scenes of what happens at the back after an image has been selected or deselected as target. Each function involves dealing with the environment variables *TARGET_IMAGE*, the image itself, and *IMAGE_LIST*, a text file containing list of images used to feed the Image Retrieval model. Furthermore, if the target image is deleted from the image list, deselection is triggered as a way to reset the image list and leave the target spot available for the rest. It is much simpler to delete the file and regenerate another one instead of a whole process of opening and editing it with Python.

5.6 User Interface

When first accessed or got refreshed, Flask renders the bare HTML template containing a title, image table column names with no image yet, choices of the number of results to display, and some buttons to function the uploading of images and trigger the image retrieval model inference. After extracted from the request body, the list is looped through in such a way that for every row that is loaded to the table, there comes along with an image, date of the upload, and actions for the item to be either deleted or selected/deselected as target image.

After images are present on the website, each of them has a fair chance to be picked as the target and compared with the others for close matches. The predict button is initially set to be disabled, and it remains in its state until an image is selected, as to prevent frontend from providing missing arguments to the image retrieval model in the backend. Another small and not-so-different section, yet worth mentioning, is the radio buttons to decide how many results users want the application to display for results, also known as *k*, or *top-k* results, and there are choices of 1, 3, and 5 now. These number of result images are considered sufficient for users to observe the overall performance as each element of image takes up quite much space. By default, *k* is always set to 1 meaning to just pick the photo closest to the target image, and that is usually the case for most users. However, another reason for this default setting of *k* is to back up the case that users forget to choose with a *top-k* value determined already, so that the model always gets a full set of arguments to proceed, and the web application has the information to list at least 1 result image.

When the "Predict" button is enabled, it is a sign that you are good to go, and the model is ready to be utilized. After the button is triggered, all the information needed is collected into a set of arguments and passed to the module in the backend for image processing and comparison. The processing speed depends on the number of images there are in the list. The longer the list, the longer it takes for the model to produce answers. The sign when result is up is that target image is shown once again underneath the "Predict" button, and then comes the result list. Each result comprises an image, and a quotient indicating the distance between it and the target, as the smaller the distance, the closer the pair. These rows are arranged in ascending order of distance so as to rank the images from closest to furthest.

Last but not least, there is a button titled "Refresh" on top of the query image list. As the name refers, this button is used to refresh the whole application and return to its original state. When "Refresh" is clicked, what happens is that all the information about the query images, target image, or results are erased, meaning that they are all set back to their default values, and the user interface redirects to its initial display, which users see when the app pops up in the first access.

With this button, users manage to erase everything present on the website with just a simple click, and it can even remove the whole predictions section in the cleanest way possible while deleting every item on the list single-handedly neither hides the whole prediction section nor presents the images in the prediction list since the image sources are nowhere to be found.

6 DISCUSSION AND FURTHER DEVELOPMENT

The goal of this project was to create a platform where people could upload any collection of images of their choice and were able to pick out those that most closely matched the target image. After a period of researching and working on the application, what was acquired has achieved the desired objectives:

- Broadening knowledge in the field of image retrieval, with its best technique up to now CBIR, and how it works.
- Structuring the application design with a general pipeline for integrating a machine learning model into a web application.
- Learning about transfer learning, how different it is from other learning types, and how a project is able to apply transfer learning to an application pipeline, instead of training a model from scratch.

Research on image retrieval was done and the web application was successfully built. Later, the project was put to the test and the results were shown in APPENDIX 1. The application completed what it was assigned when the final retrieved image was the exact same as the target image, selected among the image list, as well as ranking from closest to furthest according to distances between them. Besides, the API endpoints appeared to function properly to transmit information from frontend to backend, and the inference predictions vice versa.

Nevertheless, implementations encountered limitations as flaws and inaccuracies were revealed under performance testing. In APPENDIX 1, it was pointed out that the model only worked best when the number of images was low, and pixel features needed to be somewhat distinct between the images. However, when there was a small increase in the quantity of images, the model tended to fall off the beaten track and made the wrong decisions, hence irrelevant retrievals. The reason for this was that a huge image set caused the model to be confused anticipated predictions with other items. Moreover, the more images users uploaded onto the app, the longer it took the model to run a complete process toward final answers.

From the problems discussed above, there is always room for improvement. In the aspect of machine learning, if given enough time, and powerful computing resources, it is possible to obtain a better-performed image retrieval model that is trained from scratch, tested, and well-evaluated within multiple iterations. Web development needs enhancements as well. The app requires implementation that can assist in dealing with POST requests and processing the images faster. In addition, the images should be arranged differently, possibly in grids, so that scrolling to the result section underneath remains as few as there can be.

REFERENCES

Altbach, P. & de Wit, H. 2018. Too much academic research is published. Read on 12.02.2023.
https://www.universityworldnews.com/post.php?story=20180905095203579

arXiv.org blog. 2022. Two million articles and counting!. Read on 08.02.2023.
https://blog.arxiv.org/2022/01/03/two-million-articles-and-counting/

arXiv info. no date. arXiv submission rate statistics. Read on 08.02.2023.
https://info.arxiv.org/help/stats/2021_by_area/index.html

Baeldung. no date. What is Content-based Image Retrieval?. Updated on 10.12.2022. Read on 20.02.2023.
https://www.baeldung.com/cs/cbir-tbir

Birchard, T. 2018. Configuring Your Flask App. Read on 19.01.2023.
https://hackersandslackers.com/configure-flask-applications/

CERN. no date. A short history of the Web. Read on 25.02.2023.
https://home.cern/science/computing/birth-web/short-history-web

Condon, S. 2018. Intel launches toolkit to bring computer vision to the edge. Read on 19.01.2023
https://www.zdnet.com/article/intel-launches-toolkit-to-bring-computer-vision-to-the-edge/

EDUCBA. no date. Flask Environment Variables. Read on 12.01.2023.
https://www.educba.com/flask-environment-variables/

Fireship. 2021. Python in 100 Seconds. Read on 28.12.2022.
https://www.youtube.com/watch?v=x7X9w_GIm1s

Goodfellow, I., Bengio, Y. & Courville, A. 2016. Deep learning. MIT press. Read on 15.03.2023.

https://mitpress.mit.edu/9780262035613/deep-learning/

Grinberg, M. 2018. Flask Web Development, 2nd Edition. Read on 28.12.2022. https://www.oreilly.com/library/view/flask-web-development/9781491991725/

Intel. no date. Image Retrieval Python* Demo. Read on 10.01.2023. https://docs.openvino.ai/latest/omz_demos_image_retrieval_demo_python.html

Intel. no date. image-retrieval-0001. Read on 10.01.2023. https://docs.openvino.ai/2021.1/omz_models_intel_image_retrieval_0001_description_image_retrieval_0001.html

Intel. no date. Overview — OpenVINO™ documentation — Version(latest). Read on 19.01.2023. https://docs.openvino.ai/latest/home.html

Jayaram, S. 2017. Integrating a Machine Learning Model into a Web app. Read on 12.03.2023. https://github.com/shivasj/Integrating-a-Machine-Learning-Model-into-a-Web-app

MacManus, R. 2020. 1995: The Birth of Javascript. Read on 25.02.2023. https://webdevelopmenthistory.com/1995-the-birth-of-javascript/

Marcotte, E. 2011. Responsive Web Design. Read on 25.02.2023. https://abookapart.com/products/responsive-web-design

Mozillia. no date. JavaScript Guide. Read on 28.12.2022. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. 2014. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Vattern Recognition, 1717-1724. Read on 11.03.2023. https://ieeexplore.ieee.org/document/6909618

OpenVINO™ Toolkit. 2019. Image Retrieval Python* Demo. Read on 10.01.2023. https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos/image_retrieval_demo/python

Pan, S. J., & Yang, Q. 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22 (10), 1345-1359. Read on 12.03.2023. https://ieeexplore.ieee.org/document/5288526

Python. no date. About Python | Python.org. Read on 28.12.2022. https://www.python.org/about/

Red Hat. 2020. What is a REST API?. Read on 14.03.2023. https://www.redhat.com/en/topics/api/what-is-a-rest-api

Sato, D., Wider, A. & Windheuser, C. 2019. Continuous Delivery for Machine Learning. Read on 12.03.2023. https://martinfowler.com/articles/cd4ml.html

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A. & Jain, R. 2000. Content-Based Image Retrieval at the End of the Early Years. IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12), 1349-1380. Read on 15.03.2023. https://www.researchgate.net/publication/3193201_Content-based_image_retrieval_at_the_end_of_the_early

Sports Interactive. 2022. Football Manager 2023. Read on 18.03.2023. https://www.sigames.com/games/football-manager-2023

StackOverflow. 2019. Uploading multiple files with Flask. Read on 11.01.2023. https://stackoverflow.com/questions/11817182/uploading-multiple-files-with-flask

TCM Logos. 2022. TCM23 – Logopack FM23 / FM2023. Updated on 02.01.2023. Read on 18.03.2023.
https://www.tcmlogos.com/tcm23-logos-fm23/

The Pallets Projects. no date. Introduction - Jinja Documentation (3.1.x). Read on 19.01.2023.
https://jinja.palletsprojects.com/en/3.1.x/intro/

The Pallets Projects. no date. Quickstart - Flask Documentation (2.2.x). Read on 11.01.2023.
https://flask.palletsprojects.com/en/2.2.x/quickstart/#rendering-templates

The Pallets Projects. no date. Template Designer Documentation - Jinja Documentation (3.1.x). Read on 19.01.2023.
https://jinja.palletsprojects.com/en/3.1.x/templates/#template-inheritance

The Pallets Projects. no date. Uploading Files - Flask Documentation (2.2.x). Read on 11.01.2023.
https://flask.palletsprojects.com/en/2.2.x/patterns/fileuploads/

The Pallets Projects. no date. Welcome to Flask - Flask Documentation (2.2.x). Read on 28.12.2022.
https://flask.palletsprojects.com/en/2.2.x/

U.S. Bureau of Labor Statistics. no date. Web Developers and Digital Designers. Updated on 14.09.2022. Read on 25.02.2023.
https://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm

Weiss, K., Khoshgoftaar, T. M., & Wang, D. 2016. A Survey of Transfer Learning. Journal of Big Data 3 (1), 1-40. Read on 15.03.2023.
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6

White, M. 2022. Football Manager 2023: Everything we know about FM23, including best price, new features, wonderkids and more. Read on 18.03.2023.

https://www.fourfourtwo.com/features/football-manager-2023-everything-we-know-about-fm23-including-preorders-new-features-release-date-and-more

Wikipedia. no date. Bamboutos FC. Updated on 01.10.2022. Read on 18.03.2023.
https://en.wikipedia.org/wiki/Bamboutos_FC

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. 2014. How transferable are features in deep neural networks?. In Advances in neural information processing systems, 3320-3328. Read on 14.03.2023.
https://arxiv.org/abs/1411.1792

APPENDICES

Appendix 1. Football Manager 2023 Team Logos Experiment

This appendix is dedicated to documenting how well the web-based image retrieval application performs. The optimal approach to testing its capabilities is to apply it to a particular task. In this case, the task was associated with Football Manager 2023 (FM 23), a well-known football management simulation game.

FM 23, developed by Sports Interactive and released in 2022, is a well-known sports simulation video game that allows players to take over the management of a football club and navigate various aspects of running a team, from training and tactics to transactions and finances. The Football Manager series has long been welcomed and enjoyed by fans for its updated features, improved gameplay, and new challenges. The only downside of the game is that not all football clubs have their official logos in the gameplay. Instead, Sports Interactive replace them with their designs. In fact, in the Premier League, one of the most prestigious league divisions in the world, there are only a few teams, like Manchester City (White, 2022) or Watford, carrying the true logo. The fact that whether the logo displayed in the game is genuine or not depends on partnerships between football clubs and Sports Interactive, and it seems reasonable to only display those that are in partnerships with the game developers.

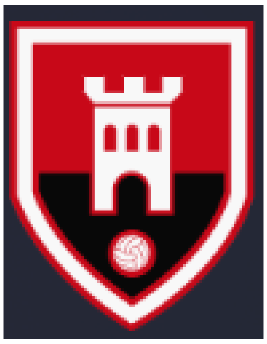


FIGURE 9: Manchester United logo in Football Manager 2023 (Sports Interactive, 2022)

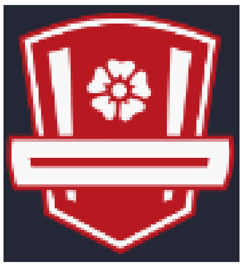
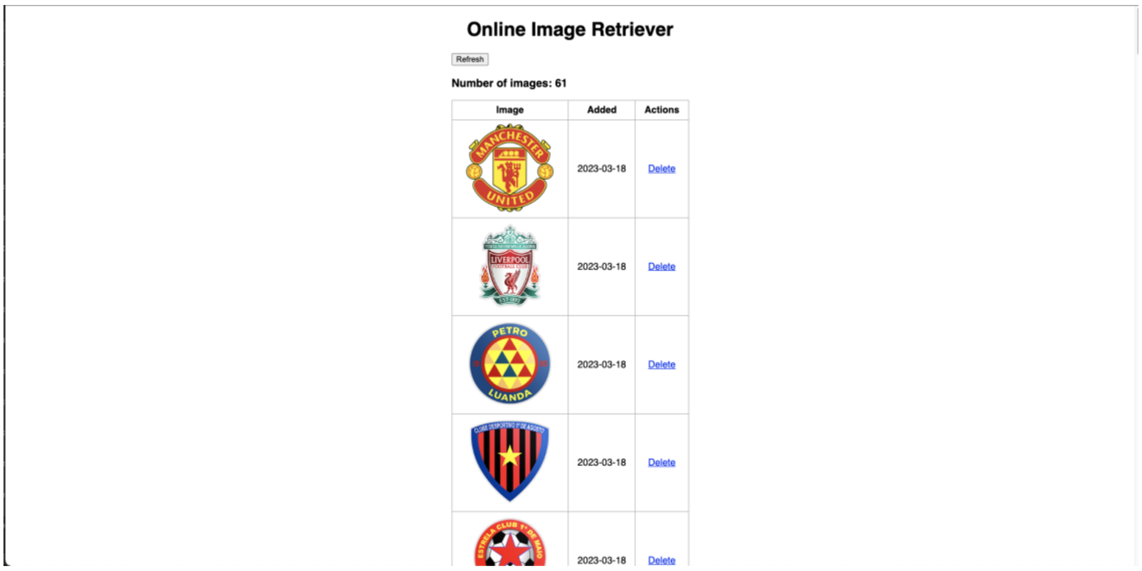


FIGURE 10: Liverpool logo in Football Manager 2023 (Sports Interactive, 2022)

As a fan, it is always more fun and exciting to experience the game with the latest updates on football teams, players, transfers, graphics, and so much more. On the Internet, there are numerous logo packs designed and published by fans of the series. This megapack includes logos for every single club, league, and competition throughout regions and levels. Each item in the pack contains a serial number and it is directly indicated in the filename. Moreover, in order to successfully insert the pack into the game, pairs of images need to be matched from the pack and the source directory of the game, via the mutual serial number.

That said, this was an excellent opportunity to put the image retriever to the test. The goal was to figure out the filename of the official logo of Liverpool football club. The target image was an image of the logo of Bamboutos FC, a football team based in Mbouda, Cameroon (Wikipedia, no date), outside the FM 23 megapack, and an image set included logos of Machester United, Liverpool, and those from Africa.

In this experiment, there were two test cases: image retrieval with a small dataset, around 50-60 images, and a slightly bigger set of approximately 200 images.



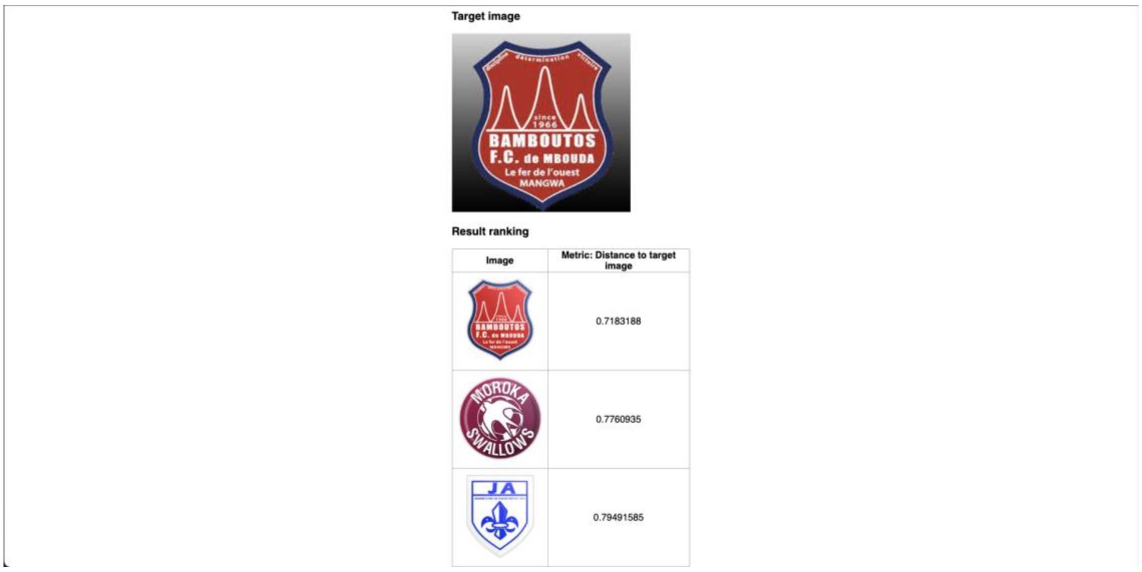
Online Image Retriever

Refresh

Number of images: 61

Image	Added	Actions
	2023-03-18	Delete
	2023-03-18	Delete
	2023-03-18	Delete
	2023-03-18	Delete
	2023-03-18	Delete

FIGURE 11: Application interface with total number of images.

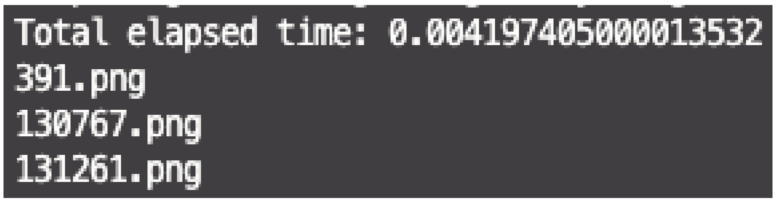


Target image

Result ranking

Image	Metric: Distance to target image
	0.7183188
	0.7760935
	0.79491585

FIGURE 12: Result section of the application in the first experiment.



```
Total elapsed time: 0.004197405000013532
391.png
130767.png
131261.png
```

FIGURE 13: Filename result of the first experiment.

In the first example, it was apparent that the model retrieved the desired image as its top 1 contender effortlessly, with its distance being the least towards the query image, and the serial number needed was 391.

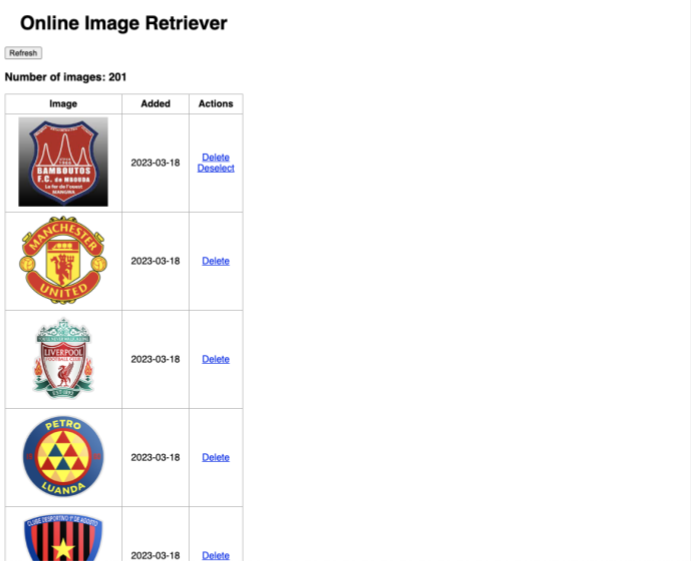


FIGURE 14: Application interface with total number of images.

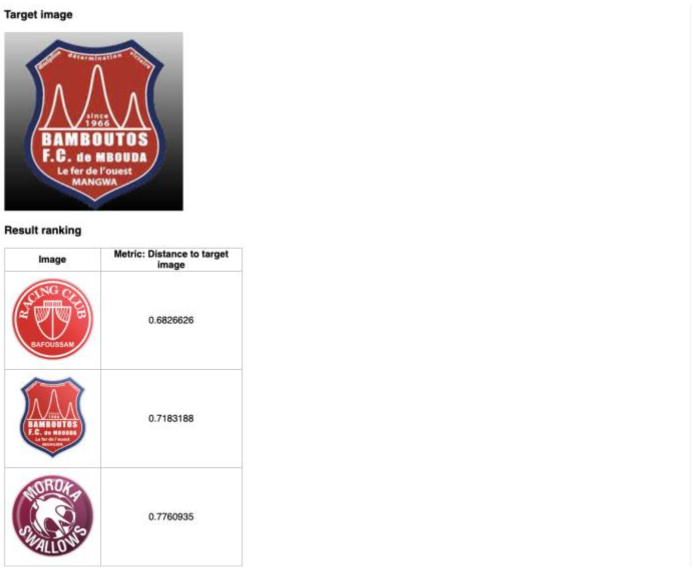
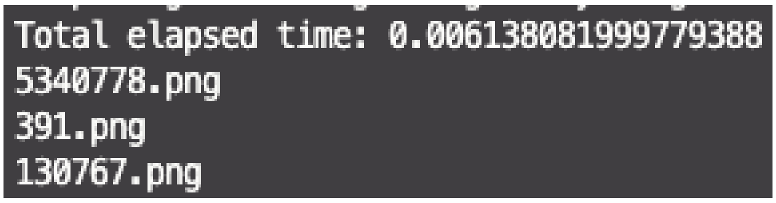


Image	Metric: Distance to target image
	0.6826626
	0.7183188
	0.7760935

FIGURE 15: Result section of the application in the second experiment.



```
Total elapsed time: 0.006138081999779388
5340778.png
391.png
130767.png
```

FIGURE 16: Filename result of the second experiment.

When there was a small change in the total number of images, as the model processed and took more images into consideration, it tended to get more confused with all the features, colors or shapes down to pixel level. This combined would later cause the model to make the wrong decisions. Even though the desired item was in second place, the model still needs improving to perform better on other logos or service other tasks more effectively as well.