Image Search Engine with VGG16, Elasticsearch, Flask, and Docker

Kaouech Mohamed & Angela Ben Frej

December 25, 2024

Abstract

This report describes the design, implementation, and features of an image search engine system that leverages the VGG16 deep learning model for feature extraction, Elasticsearch for indexing, Flask for the backend API, and Docker for containerization. The system is designed to efficiently search and retrieve images based on visual similarity using advanced techniques such as deep learning and fast data indexing.

1 Introduction

In the modern digital world, image search engines are essential tools for retrieving visually similar images based on input queries. This project aims to build a robust image search engine that uses deep learning for feature extraction and Elasticsearch for efficient and scalable indexing and searching of images.

The system involves several core components:

- **Image Similarity Search**: Upload an image and find visually similar images.
- **VGG16-based Feature Extraction**: Extracts detailed feature vectors for accurate search results.
- **Flask API**: Fast and reliable API to process search requests.
- **Interactive Web Interface**: Easy-to-use frontend built with HTML, CSS, and JavaScript.
- **Dockerized Services**: The project is fully containerized, making it easy to deploy and scale.
- **Refined Search**: After performing an initial search, users can click on any displayed image to search again for similar images based on the selected image. This allows users to drill down further into their search results and explore related content.

This report outlines the architecture, technology stack, and implementation details of the system.

1.1 User Interface Example

Here is a screenshot of the user interface where users can upload images and perform similarity-based searches.

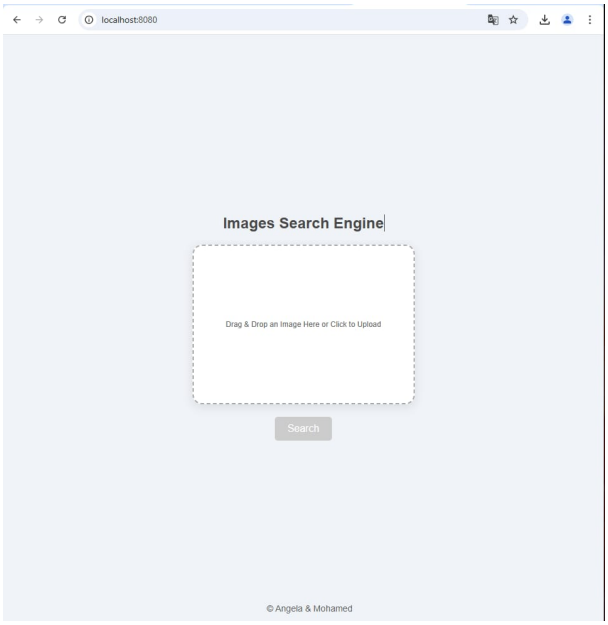

Figure 1: User Interface of the Image Search Engine

2 Project Architecture

The architecture of the image search engine consists of multiple components working together to provide a seamless user experience for image retrieval.

2.1 VGG16 Feature Extraction

The VGG16 model is a convolutional neural network (CNN) that has been pre-trained on large image datasets. It is used in this project to extract high-level feature vectors from images. These vectors capture key characteristics of the image such as texture, shape, and color, making them ideal for similarity-based searches.

The VGG16 model is used to process each image in the dataset and produce a corresponding feature vector, which is then indexed in Elasticsearch. These feature vectors serve as the basis for image similarity search.

2.2 Elasticsearch Integration

Elasticsearch is a highly scalable and distributed search engine that is widely used for fast and efficient text-based search operations. In this project, Elasticsearch is used to index the feature vectors extracted by the VGG16 model.

The key advantage of using Elasticsearch is its ability to perform similarity-based search in near real-time. Once the images are indexed, users can search for images by either uploading a query image or entering search keywords. Elasticsearch finds the most similar images based on the feature vectors, ensuring fast and accurate search results.

2.3 Flask Backend API

Flask is a lightweight Python framework that is used to build the backend of the image search engine. The Flask API is responsible for handling user requests, including image uploads and search queries.

Upon receiving a query, the backend API processes the image using the VGG16 model to extract the feature vector and searches the Elasticsearch index for similar images. The API then returns the results to the frontend, which displays them to the user.

2.4 Frontend Interface

The frontend of the image search engine provides a simple and interactive user interface for performing searches. Users can either upload an image or enter text-based search queries. The interface is built using HTML, CSS, and JavaScript, providing a responsive and easy-to-use platform for users to interact with the system.

2.5 Dockerization

To ensure consistent deployment and scalability, the entire system is containerized using Docker. Docker allows each component (frontend, backend, and Elasticsearch) to be packaged in its own container. This ensures that the system works seamlessly across different environments and can be easily scaled when needed.

Docker Compose is used to manage and orchestrate the multiple containers, making it simple to deploy the entire application with a single command.

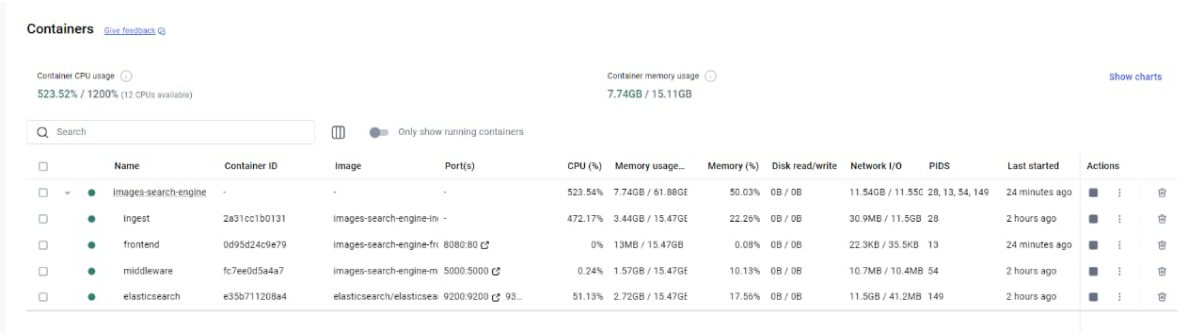

Figure 2: Containers

3 System Workflow

The system operates in the following manner:

1. **Image Upload**: The user uploads an image or enters a search query in the frontend interface.

2. **Feature Extraction**: The Flask backend processes the image using the VGG16 model, which converts the image into a feature vector.

3. **Search Query**: The feature vector is sent to Elasticsearch, which indexes the vectors and performs a similarity-based search to find the closest images.

4. **Search Results**: The Flask API retrieves the results from Elasticsearch and sends them back to the frontend.

5. **Display Results**: The frontend displays the similar images to the user, completing the search cycle.

4 Technology Stack

The system is built using the following technologies:

- **VGG16**: A pre-trained convolutional neural network used for feature extraction.
- **Elasticsearch**: A distributed search engine for indexing and querying image features.
- **Flask**: A lightweight Python framework for building the backend API.
- **HTML, CSS, JavaScript**: Technologies used for building the frontend interface.
- **Docker and Docker Compose**: Tools used for containerizing and orchestrating the application services.

5 Implementation Details

5.1 VGG16 Feature Extraction

VGG16 is a deep convolutional neural network with 16 layers that has been trained on the ImageNet dataset. In this project, the model is used to extract feature vectors from images by passing them through the network. These vectors are then normalized and stored for similarity search.

5.2 Elasticsearch Indexing

After extracting the feature vectors, they are indexed in Elasticsearch. The index stores the feature vectors along with metadata about the images, such as their filenames and tags. Elasticsearch allows for efficient similarity-based searches, making it ideal for this project.

5.3 Flask API

The Flask API is designed to handle various requests:

- Image upload for feature extraction.
- Retrieval of similar images based on uploaded images or feature vectors.

5.4 Refined Search Functionality

After performing an initial search and displaying the top similar images, users have the ability to select any displayed image to search for more images similar to it. This refined search option allows users to drill deeper into the search results by selecting images that appear most relevant to their needs. When an image is clicked, the backend performs another feature extraction based on the selected image and updates the search results accordingly.

5.5 Dockerization

The project is fully containerized using Docker, and Docker Compose is used to orchestrate the containers. The following services are containerized:

- Frontend service (handles user interactions).
- Flask API service (handles backend processing).
- Elasticsearch service (handles indexing and searching).

6 Results and Evaluation

The image search engine successfully retrieves visually similar images based on the feature vectors extracted by the VGG16 model. The Elasticsearch index allows for fast and efficient searches, and the Flask API ensures that the system can handle a large number of requests. The user interface is simple and intuitive, allowing users to interact with the system easily. And this is an example of an image Search

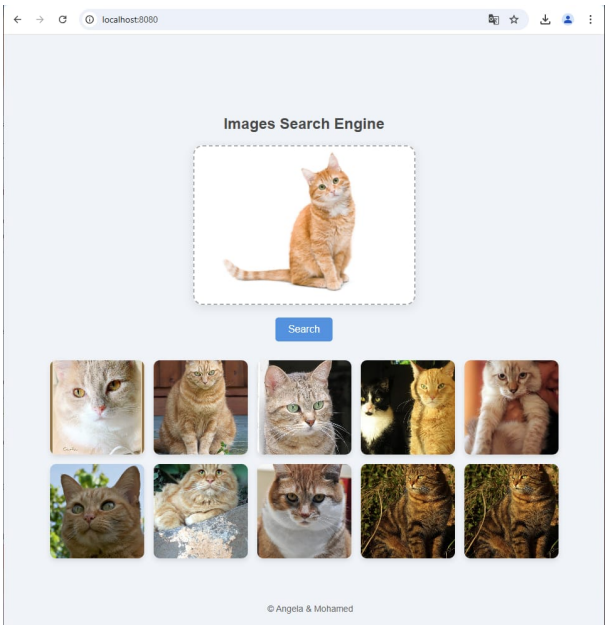

Figure 3: Testing the image search engine

7 Conclusion

This project demonstrates the power of combining deep learning, search engines, and web frameworks to build a high-performance image search engine. By using VGG16 for feature extraction and Elasticsearch for efficient indexing and searching, the system can handle large-scale image retrieval tasks. The Dockerized architecture ensures that the system is scalable and easy to deploy in various environments.

`https://github.com/KaouechMohamed/Images-Search-Engine`