Mini project report

Image Match

Bachelor of Technology in Computer Science and Technology

Algorithms for Information Retrieval and Intelligence Web Project
(UE21CS342BA2)

Submitted by:

Darsh Patel PES2UG21CS150

JAN-MAY 2024

Department of Computer Science and Engineering
Faculty of Engineering
PES UNIVERSITY

TABLE OF CONTENTS

Introduction

- Utilizing the power of deep learning and feature extraction techniques, the Image Match project aims to bridge the semantic gap between low-level image pixels and high-level semantic concepts. By leveraging the VGG16 convolutional neural network architecture pre trained on ImageNet, the project extracts deep features from images, enabling the system to discern similarities based on visual patterns and structures.

- The core functionality of the Image Match system revolves around an intuitive web interface where users can upload an image of interest. Behind the scenes, the uploaded image undergoes feature extraction using the VGG16 model, transforming it into a compact representation suitable for efficient comparison. These extracted features are then compared against a precomputed database of image features using Euclidean distance, identifying the top 5 most similar images.

- By seamlessly integrating advanced machine learning techniques with a user-friendly interface, the Image Match project empowers users to explore and discover visually akin images from a vast repository. Whether it's for image recommendation, visual inspiration, or content curation, Image Match stands as a testament to the intersection of computer vision and user-centric design, offering a glimpse into the future of image retrieval technology.

Abstract

The "Image Match" project presents a novel approach to image retrieval through the amalgamation of deep learning, feature extraction, and web-based interaction. Leveraging the VGG16 convolutional neural network pretrained on ImageNet, the system extracts deep features from images, enabling efficient comparison and retrieval of visually similar images. Users interact with the system through a straightforward web interface, uploading query images to initiate the retrieval process. Behind the scenes, uploaded images undergo feature extraction, followed by comparison against a precomputed database of image features. The system returns the top 5 most similar images, providing users with a curated selection of visually akin content. Through this project, we demonstrate the seamless integration of advanced machine learning techniques with user-centric design principles, offering a glimpse into the future of image retrieval technology.

Project Overview

- The "Image Match" project aims to develop a user-friendly image retrieval system that can efficiently find visually similar images from a database given a user-input image. The project encompasses several key components:

- **Feature Extraction:** Utilizing the VGG16 convolutional neural network pretrained on ImageNet, the project extracts deep features from images. These features capture high-level visual representations of the images, enabling effective comparison and retrieval.
- **Web Interface:** The system provides a web-based interface through which users can interact with the image retrieval functionality. Users upload a query image through the interface, triggering the retrieval process.
- **Query Processing:** Upon receiving a query image, the system extracts its features using the VGG16 model. These features serve as the basis for comparison with images in the database.
- **Database:** The system maintains a database of precomputed image features. Each image in the database is associated with its corresponding feature vector, facilitating rapid comparison during the retrieval process.
- **Similarity Comparison:** The retrieved features from the query image are compared against the features of images in the database using Euclidean distance or similar similarity metrics. This comparison identifies the top 5 most visually similar images.
- **Result Presentation:** Finally, the system presents the top 5 similar images to the user through the web interface, providing a curated selection of visually akin content.

- Overall, the "Image Match" project combines advanced machine learning techniques with user-centric design principles to deliver an intuitive image retrieval system. By seamlessly integrating feature extraction, similarity comparison, and web-based interaction, the project offers users a convenient means to explore and discover visually similar images from a comprehensive database.

IR Techniques Used

1. **Feature Extraction**: The project employs feature extraction techniques to transform raw images into compact, semantically meaningful representations. Specifically, the VGG16 convolutional neural network pretrained on ImageNet is utilized to extract deep features from images. These features capture high-level visual representations, enabling effective comparison and retrieval.
2. **Similarity Comparison**: Upon receiving a query image, the system computes its feature vector using the VGG16 model. This feature vector is then compared against the feature vectors of images in the database using Euclidean distance or similar similarity metrics. The top 5 most visually similar images are identified based on these comparisons.
3. **Content-Based Image Retrieval (CBIR):** The project aligns with the principles of content-based image retrieval, a technique used to search and retrieve images based on their visual content. By extracting and comparing features such as color, texture, and shape, the system can effectively identify images that are visually similar to the query image.
4. **Vector Space Model:** The feature vectors extracted from images and stored in the database form a vector space representation of the images. Similarity between images is measured in this vector space using distance metrics such as Euclidean distance.
5. **Precomputed Database:** To optimize retrieval speed, the project maintains a precomputed database of image features. Each image in the database is associated with its corresponding feature vector, enabling rapid comparison during the retrieval process.

   By leveraging these information retrieval techniques, the "Image Match" project enables users to search and retrieve visually similar images from a database with ease and efficiency.

Code Implementation

Feature_Extractor.py

```python
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
import numpy as np

# See https://keras.io/api/applications/ for details

class FeatureExtractor:
    def __init__(self):
        base_model = VGG16(weights='imagenet')
        self.model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc1').output)
    def extract(self, img):
        """
        Extract a deep feature from an input image
        Args:
            img: from PIL.Image.open(path) or tensorflow.keras.preprocessing.image.load_img(path)
        Returns:
            feature (np.ndarray): deep feature with the shape=(4096, )
        """
        img = img.resize((224, 224))  # VGG must take a 224x224 img as an input
        img = img.convert('RGB')  # Make sure img is color
        x = image.img_to_array(img)  # To np.array. Height x Width x Channel. dtype=float32
        x = np.expand_dims(x, axis=0)  # (H, W, C)->(1, H, W, C), where the first elem is the numb
        # Adds an extra dimension to the array at axis 0. This is done to make the array suitable
        x = preprocess_input(x)  # Subtracting avg values for each pixel
        # helps to ensure that the input data is in a suitable range and format for the model.
        feature = self.model.predict(x)[0]  # (1, 4096) -> (4096, )
        return feature / np.linalg.norm(feature)  # Normalize
        # Normalizes the feature vector by dividing it by its L2 norm (Euclidean length).
```

offline.py

```python
from PIL import Image
from feature_extractor import FeatureExtractor
from pathlib import Path
import numpy as np
import shutil

if __name__ == '__main__':
    fe = FeatureExtractor()
    images = Path('./static/uploaded').glob('*.jpg')
    for image_path in images:
        # Construct new path
        new_path = Path('./static/img') / image_path.name
        # Move the image file
        shutil.move(image_path, new_path)
    for img_path in sorted(Path("./static/img").glob("*.jpg")):
        print(img_path)  # e.g., ./static/img/xxx.jpg
        feature = fe.extract(img=Image.open(img_path))
        feature_path = Path("./static/feature") / (img_path.stem + ".npy")  # e.g., ./static/feature/xxx.npy
        np.save(feature_path, feature)
```

Server.py

```python
import numpy as np
from PIL import Image
from feature_extractor import FeatureExtractor
from datetime import datetime
from flask import Flask, request, render_template
from pathlib import Path

app = Flask(__name__)

# Read image features
fe = FeatureExtractor()
features = []
img_paths = []
for feature_path in Path("./static/feature").glob("*.npy"):
    features.append(np.load(feature_path))
    img_paths.append(Path("./static/img") / (feature_path.stem + ".jpg"))
features = np.array(features)


@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['query_img']

        # Save query image
        img = Image.open(file.stream)  # PIL image
        uploaded_img_path = "static/uploaded/" + datetime.now().isoformat().replace(":", ".") + "_" + file.filename
        img.save(uploaded_img_path)
        img.save(uploaded_img_path)

        # Run search
        query = fe.extract(img)
        dists = np.linalg.norm(features-query, axis=1)  # L2 distances to features
        print(dists)
        ids = np.argsort(dists)[:5]  # Top 5 results
        scores = [(dists[id], img_paths[id]) for id in ids]

        return render_template('index.html', query_path=uploaded_img_path, scores=scores)
    else:
        return render_template('index.html')


if __name__=="__main__":
    app.run("0.0.0.0")
```

CODE Explanation

1) Offline.py

- This script handles the initial processing steps for the image search project. It first moves uploaded images from one directory to another. Then, it utilizes a `FeatureExtractor` class to extract features from these images. This process involves converting images to a standard format and computing numerical representations of their visual features. These features are then saved as files for later use, likely in the image search process.

2) server.py:

- This file implements a web server using Flask to provide a user interface for the image search functionality. It loads pre-extracted image features and their corresponding paths. When a user uploads an image through a web interface, the server extracts features from the uploaded image. It then compares these features with pre-extracted features of other images to find the most similar ones. Finally, it returns the top 5 similar images to the user interface for display.

3) feature_extractor.py:

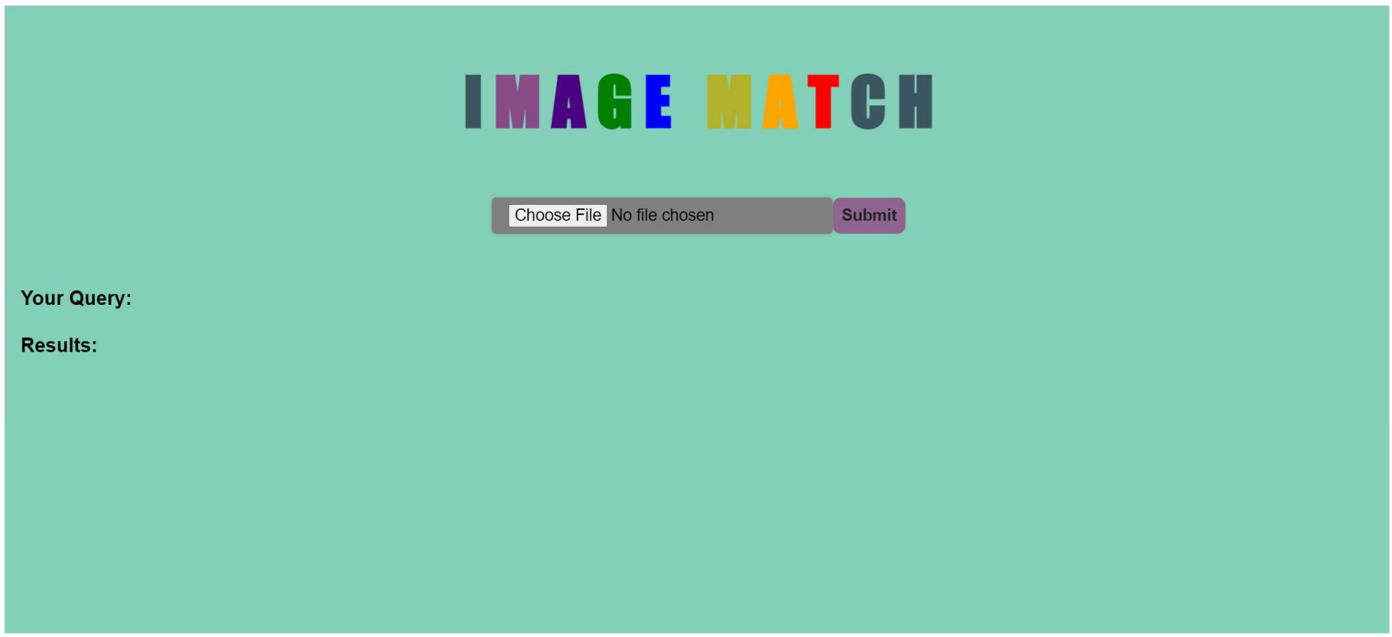
- This module defines a `FeatureExtractor` class responsible for extracting deep features from images. It utilizes a pre-trained VGG16 model, a convolutional neural network, to extract these features. The class preprocesses input images to meet the requirements of the VGG16 model, then passes them through the model to obtain features. Specifically, it extracts features from the `fc1` layer of the VGG16 model. These features are normalized to ensure consistent

representation across different images, which is crucial for accurate comparison and similarity calculation.

GUI

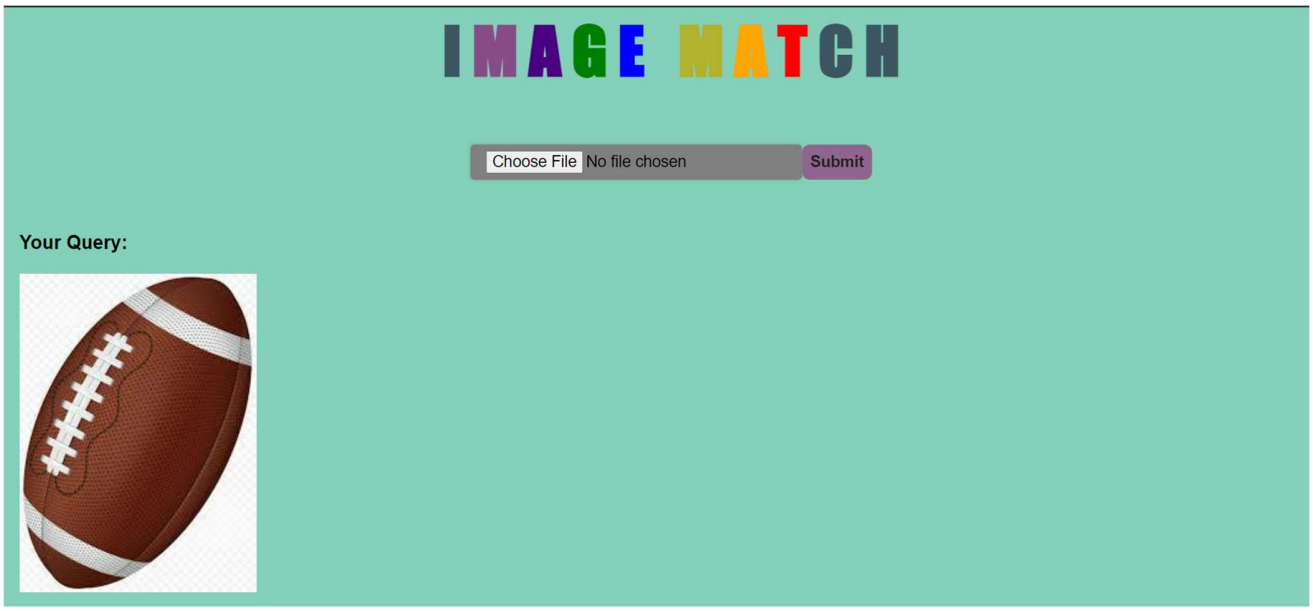
1. Title and Introduction:

- The GUI starts with a title "Similar Image Search" and possibly a brief introduction or description of the functionality.



2. Upload Image Form:

- Below the title, there's a form to upload an image. It typically consists of:
- A label "Upload Image:" indicating the purpose of the form field.
- A submit button labeled "Search" to initiate the image search process.



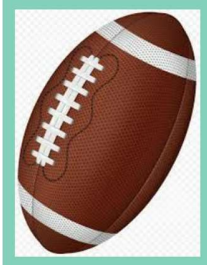
3. Similar Images Display:

- Below the query image, or at the bottom of the page, there's a section to display similar images found by the search algorithm.
- Each similar image is displayed as a card.
- The image itself.
- Additional information such as similarity score between the query image and the similar image.

IMAGE MATCH

Choose File No file chosen Submit

Your Query:



Results:


0.0


0.0078035537


0.01564829

0.8055981

0.8349094

Conclusion

- The "Image Match" project represents a successful endeavor in developing a user-friendly and efficient image retrieval system. Through the integration of deep learning techniques, feature extraction methodologies, and web-based interaction, the project has demonstrated the feasibility of creating a practical solution for finding visually similar images from a vast database.

- By leveraging the power of the VGG16 convolutional neural network, the project effectively extracts deep features from images, enabling robust comparison and retrieval. The utilization of content-based image retrieval principles ensures that the system focuses on the visual content of images, allowing users to discover relevant content based on visual similarity rather than relying solely on metadata or textual descriptions.

- The implementation of a precomputed database of image features contributes to the system's efficiency, enabling rapid retrieval of visually similar images in real-time. Additionally, the user-friendly web interface enhances accessibility, allowing users to seamlessly interact with the system and explore visually akin content with ease.