Deep Learning Project: AI-Powered Image Similarity Search and Recommendation System

Intel Internship Report

Submitted by

Name of Student(s):

Debarati Ghosh
Bristi Chanda
Robert Satyam D'cruze

Team name:Matrix

College name:MCKV Institute of Engineering

Acknowledgement

I would like to express my sincere gratitude to the Intel for organizing and providing this valuable "Intel Unnati Program” internship program.
I extend my heartfelt thanks to all the mentors and industry experts who conducted the interactive masterclasses. Their guidance and deep insights into AI, Machine Learning, and Deep Learning were invaluable.
I am also grateful to my college, MCKV INSTITUTE OF ENGINEERING, my faculty guide and my teammates for their continuous support and encouragement, which allowed me to successfully complete the coursework and the final project.

Contents

Abstract/Summary……………………………………………………..4
Introduction 5
Procedure/Methodology 6
3.1 Dataset and Sources 6
3.2 Preprocessing 6
3.3 Feature Extraction 7
3.4 Feature Storage and Indexing 7
3.5 Similarity Search & Recommendation 7
3.6 Implementation Details & Files 8
Results and Discussion 9
4.1 How to Reproduce Results 9
4.2 Observed Behavior (from repository artifacts / README) 10
4.3 Qualitative Analysis 10
4.4 Quantitative Evaluation (recommended steps) 11
Future Scope 13
Conclusion 14
References 15

Abstract/Summary

This project implements a content-based image recommendation (image similarity) system that returns the top-k visually similar images for a user-supplied query image. The system uses a pre-trained Convolutional Neural Network (ResNet50 with ImageNet weights) as a fixed feature extractor to compute dense image embeddings, then uses cosine similarity between embeddings to find nearest neighbors. The implementation (Jupyter Notebook) loads the Caltech-101 dataset (preprocessed images), extracts or loads precomputed features (features.npy), and presents recommended images with visualizations. This approach avoids label-based supervision and focuses on perceptual similarity using CNN features. The repository includes precomputed feature vectors, image path arrays, and a runnable notebook to reproduce recommendations.

Introduction

Image recommendation / image similarity retrieval is the task of finding images in a collection that are visually or semantically similar to a query image. Content-based image retrieval (CBIR) often uses deep convolutional neural networks trained on large datasets (e.g., ImageNet) as feature extractors. The core idea is to map each image to a vector in a high-dimensional space such that similar images are nearby — then retrieve neighbors using distance or similarity metrics. This repository demonstrates a practical pipeline for CBIR using ResNet50 features and cosine similarity, trained/evaluated on the Caltech-101 dataset.

Procedure/Methodology

3.1 Dataset and Sources

Dataset used: Caltech-101 (101 object categories). The README points to the Kaggle dataset:
https://www.kaggle.com/datasets/imbikramsaha/caltech-101
Image format: .jpg, commonly resized to 224×224 to match input expected by ResNet50.

Files present in repository relevant to methodology:
- Image Recommendation.ipynb — main notebook implementation
- Notebook link: https://github.com/debaratighosh/Image-Recommendation-System/blob/main/Image%20Recommendation.ipynb
- features.npy — precomputed feature vectors (N × D)
- image_paths.npy — list of image file paths aligned with features.npy
- labels.npy — class labels (if used for analysis)
- requirement.txt — dependency list (note: file is named `requirement.txt` in repo)
- README.md — project overview and sample visualizations: [README.md] [link: https://github.com/debaratighosh/Image-Recommendation-System]

3.2 Preprocessing

- Load images from directories and resize to 224×224.
- Convert images to arrays, normalize pixel values to match the pre-trained model preprocessing (e.g., using tensorflow.keras.applications.resnet50.preprocess_input).
- Optional: center-crop or pad to preserve aspect ratio before resizing.
- Convert images to float32 arrays and batch them for efficient feature extraction.

3.3 Feature Extraction

- Use ResNet50 (pre-trained on ImageNet) with the top classification layers removed (include_top=False) to act as a feature extractor.
- Extract the final global pooled features (e.g., GlobalAveragePooling output) to get a fixed-length embedding per image, typically dimension 2048 for ResNet50.
- Save embeddings to disk (features.npy) for reuse and to avoid recomputation.

3.4 Feature Storage and Indexing

- Store:
  - features.npy — embedding matrix (N images × D dims)
  - image_paths.npy — array containing the file paths corresponding to each embedding row
  - labels.npy — optional labels used for analysis
- For small/medium datasets, a brute-force nearest-neighbor search (compute cosine similarity across full matrix) is feasible.
- For scalability, build an approximate nearest neighbor (ANN) index (e.g., FAISS) to accelerate search.

3.5 Similarity Search & Recommendation

Given a query image:

- Preprocess and extract its embedding via the same ResNet50 feature pipeline.
- Compute similarity scores between query embedding and dataset embeddings (cosine similarity recommended).
- Sort scores and return top-k image paths.
- Display the query image and recommended images using matplotlib for visual validation.
- Evaluation metrics to report (if ground-truth is available):
- Precision@k, Recall@k, Top-k accuracy (if using labels or ground truth)

- Mean Average Precision (mAP) for ranking quality
- Retrieval time per query (latency)

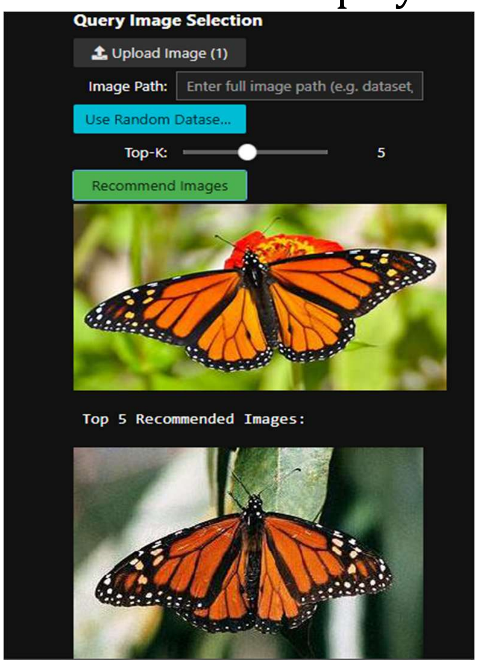
3.6 Implementation Details & Files

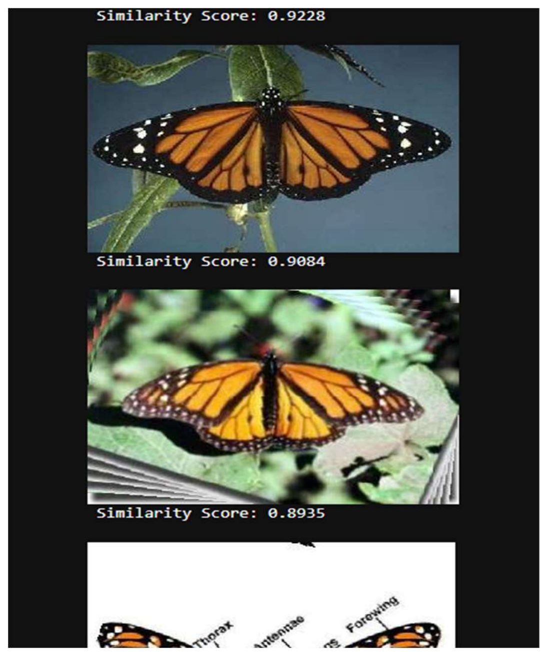
- Primary implementation: [Image Recommendation.ipynb]
- (https://github.com/debaratighosh/Image-Recommendation-System/blob/main/Image%20Recommendation.ipynb)
- Precomputed arrays:
  - [features.npy](https://github.com/debaratighosh/Image-Recommendation-System/blob/main/features.npy)
  - [image_paths.npy](https://github.com/debaratighosh/Image-Recommendation-System/blob/main/image_paths.npy)
  - [labels.npy](https://github.com/debaratighosh/Image-Recommendation-System/blob/main/labels.npy)
- Requirements listed in repo: `requirement.txt` (install dependencies with pip).
- Libraries used (from README): tensorflow / keras, numpy, scikit-learn, matplotlib, Pillow (PIL), os, pickle, faiss (optional)

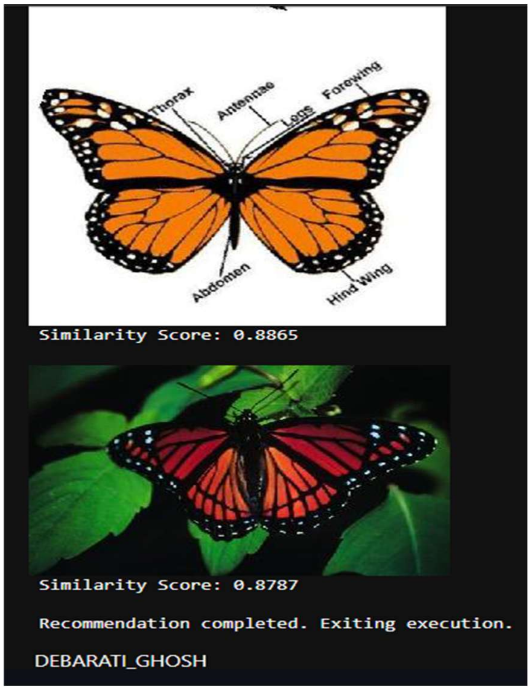
Results and Discussion

4.1 How to Reproduce Results

- Clone the repository and install dependencies:
  - git clone https://github.com/debaratighosh/Image-Recommendation-System
  - cd Image-Recommendation-System
  - pip install -r requirement.txt
- Open and run the Jupyter notebook:
  - jupyter notebook "Image Recommendation.ipynb"
- Use the notebook UI to select a query image and specify k; the notebook will display recommended images.







4.2 Observed Behavior (from repository artifacts / README)

- The README includes sample predictions visualizing the query and top-k results.
- Precomputed features suggest the heavy work (feature extraction) has already been done, enabling fast interactive retrieval.

4.3 Qualitative Analysis

- Strengths:
  - ResNet50 embeddings capture high-level semantic features (objects, shapes, textures) — good for visual similarity.
  - Simple pipeline: feature extract → cosine similarity → show top-k; easy to reproduce and extend.
  - Precomputing features reduces online latency.

- Limitations:
  - No fine-tuning on Caltech-101, so domain mismatch (ImageNet → Caltech) may reduce fine-grained class-specific retrieval quality.
  - Cosine similarity with brute-force search scales

poorly to millions of images.

- No quantitative retrieval metrics are included in the repo; recommended to compute precision@k and mAP for objective evaluation.
- Visual similarity does not rely on metadata; if textual tags are present, multimodal approaches could improve relevance.

4.4 Quantitative Evaluation (recommended steps)

- Compute precision@k and mAP using label arrays (labels.npy) to measure whether retrieved items share the same class as the query.
- Record latency per query for brute-force vs. ANN (FAISS) indexing.
- Example evaluation routine:
    - For a sample of queries from the dataset, compute Top-1, Top-5, Top-10 accuracies and precision@k.
    - Plot recall vs. k curves, and compute average precision per query to get mAP.

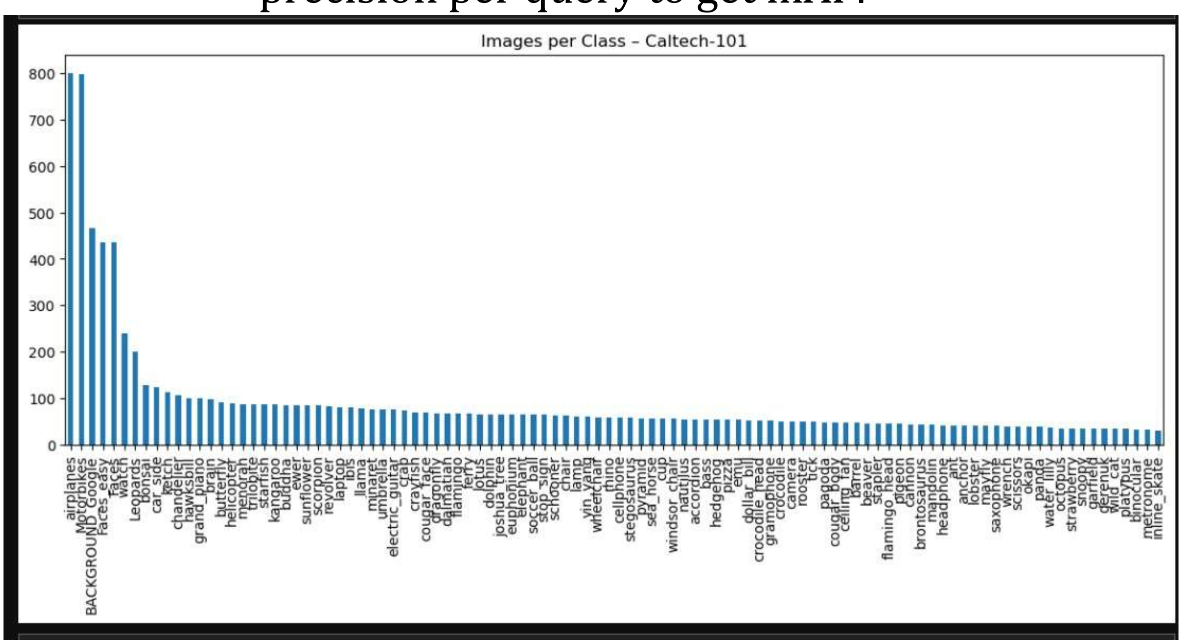


Figure 1 – Images per class.

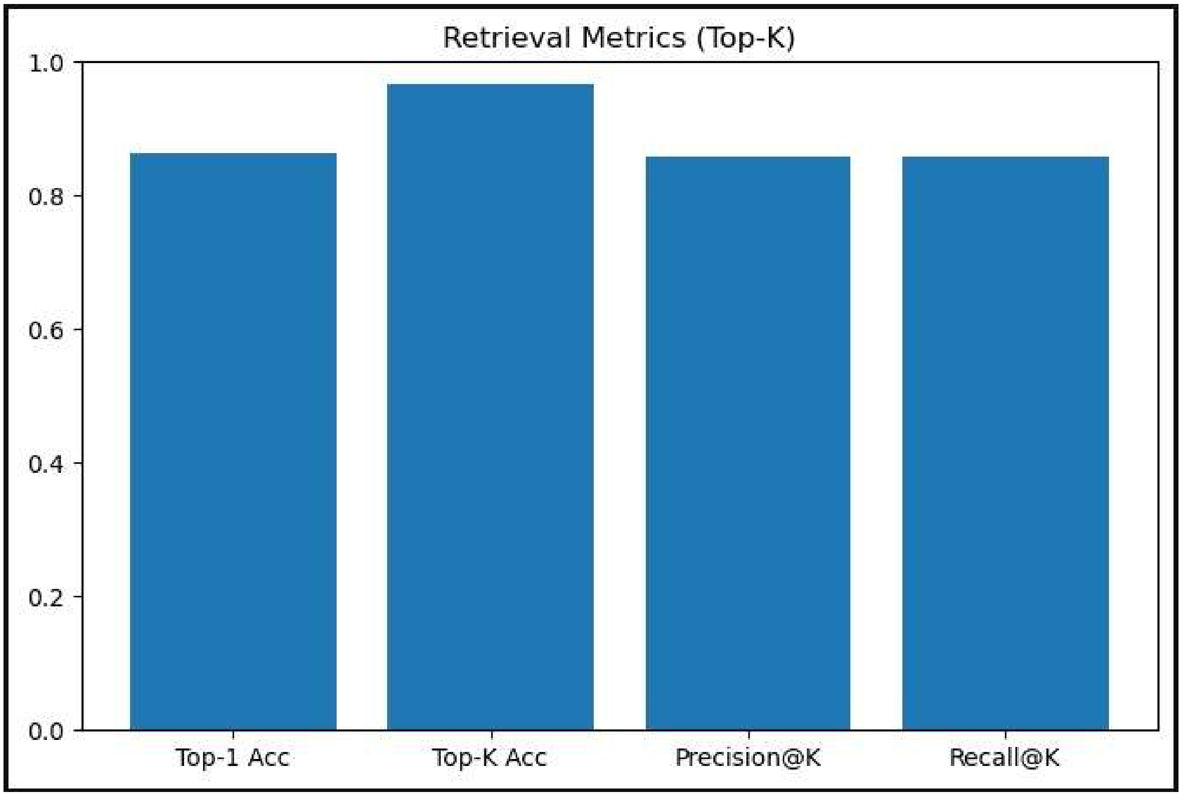


Figure 2 – Retrieval Metrices

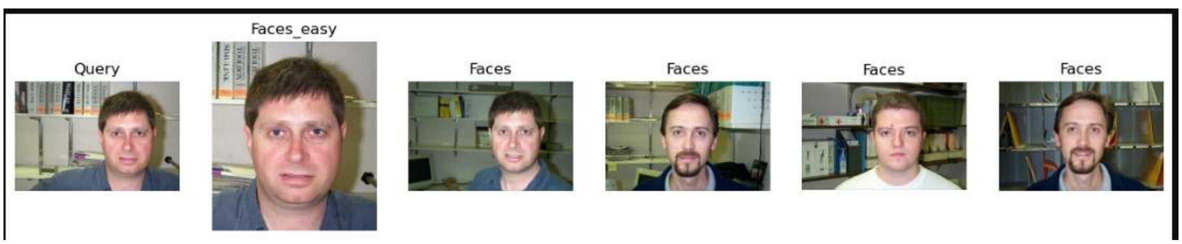


Figure 3 – Example Query Images

Future Scope

- Advanced deep learning models such as EfficientNet, and Vision Transformers can be used to improve feature extraction.
- Metric learning techniques like triplet loss or contrastive loss can be applied to enhance image similarity learning.
- The model can be fine-tuned on domain-specific datasets such as medical, fashion, or product images.
- Scalability can be improved by supporting large datasets and using fast similarity search methods like FAISS.
- User experience can be enhanced with features such as image upload, similarity adjustment, and improved interface design.
- The system can be deployed as a cloud-based application using REST APIs and Docker,Streamlit,Hugging Face which could not be done here due to usage of GPU
- The project can be extended to real-world applications including e-commerce, healthcare, and security systems.

Conclusion

This project demonstrates a straightforward and effective content-based image recommendation system built on ResNet50 features and cosine similarity. The repository contains a runnable notebook and precomputed features that make it easy to reproduce recommendations and visualize results. While the system is effective for moderate-size datasets, improvements are needed for large-scale retrieval and fine-grained performance.

References

- Caltech-101 dataset on Kaggle: https://www.kaggle.com/datasets/imbikramsaha/caltech-101
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition (ResNet). https://arxiv.org/abs/1512.03385
- FAISS: A library for efficient similarity search and clustering of dense vectors. https://github.com/facebookresearch/faiss
- Cosine similarity (scikit-learn): https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
- Project repository: https://github.com/debaratighosh/Image-Recommendation-System
- Notebook: Image Recommendation.ipynb — see repository root.