VISUAL SEARCH USING VLM

Problem Statement 7 - Intel Unnati Industrial Training

A Project Report

Submitted by

TEJAS VIPIN (RA2211028010046)

CHIRAG JAIN (RA2211028010047)

SRIJITA SAHA (RA2211028010048)

Under the Mentorship of

Dr. Annapurani. K

(Prof., Department of NWC, SRM Institute of Science and Technology)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE ENGINEERING

with specialization in CLOUD COMPUTING

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY

KATTANKULATHUR- 603 203

MARCH 2025

ABSTRACT

Large image datasets have been traditionally searched by keywords that are present in their metadata, which is often, both an unreliable and human labour and energy intensive process. We believe that, in using recent advancements in deep learning, it is now possible to extract features from an image file and then represent them as a vector. Our system uses a Vision Language Model CLIP by Open AI, to create a large dataset of image embeddings and search through them for a given query using FAISS, a highly efficient library for efficient similarity search, developed at Facebook

TABLE OF CONTENTS

INTRODUCTION

Multimedia types like images have always been quite challenging to search through for their lack of data that would make it easy for a machine to directly interpret the information presented in them. This traditionally has been solved by attaching additional information to them like text based metadata which make it easy for algorithms to search through them, but this adds a challenge of attaching accurate metadata to every image which used to largely be a manual human effort.

Using recent advancements in deep learning, it is now possible to extract features from an image and represent them as a vector. Models like CLIP correlate features in the natural language domain and the visual domain and this results in the description of an image being similar to the image itself in feature space, which allows us to easily compare and search for images.

PROPOSED PROJECT

2.1 Problem Statement

Conventional image search systems are limited by their reliance on human generated metadata and often fail in scenarios where the metadata is not descriptive enough. An ideal system would need to understand the underlying features in the image itself to function.

2.2 Objective

The primary objective of this project is to create a fully integrated system that can accept a natural language query and fetch relevant images using a search algorithm and display them to the user.

METHODOLOGY

3.1 Dataset Preparation

- **Source**: The dataset used for this project is the first 50,000 images in the *primecai/dsd_data* AI-generated image collection available on Hugging Face.
- **Structure**: Each data point includes a generated image and its corresponding descriptive text. For ease of access and scalability, these were stored in a CSV format named **“Embeddings .csv”** with three fields: Image Embedding, Text Embedding, and Description.

3.2 Vision-Language Embedding

- **Model Used**: CLIP by OpenAI.
- CLIP (Contrastive Language–Image Pretraining), a vision-language model developed by OpenAI generates 512-dimensional vector representations for both images and their corresponding text descriptions. These embeddings are aligned such that semantically similar text and images have minimal cosine distance in the embedding space, allowing for meaningful similarity comparisons.

3.3 Similarity Search

- **Tool Used**: Facebook AI Similarity Search (FAISS).
- Once the image embeddings are computed, they are indexed using FAISS (Facebook AI Similarity Search), a library optimized for large-scale similarity search and clustering of dense vectors. FAISS supports fast approximate nearest neighbor search, enabling the system to efficiently identify and retrieve the top-k most similar image vectors when provided with a textual query embedding. This significantly reduces computational overhead during real-time search queries.

3.4 Interface Design

- **Frontend Tool**: Gradio.
- In the final stage of the methodology, the entire visual search pipeline is assembled. A user inputs a natural language query, which is passed through CLIP’s text encoder to generate its embedding. These embeddings are then searched against the FAISS index to retrieve the nearest image embeddings in terms of cosine similarity. These indices are then used to fetch the images by the same name to display to the user.

IMPLEMENTATION

4.1 Architecture Diagram

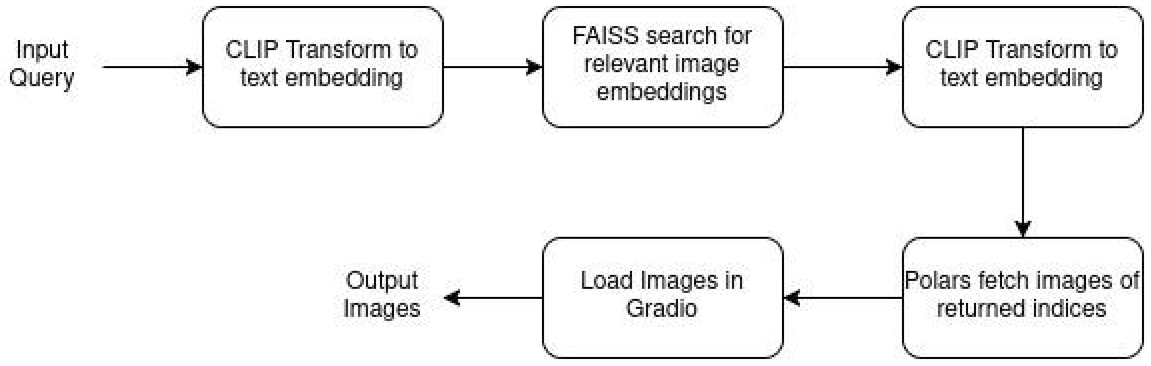
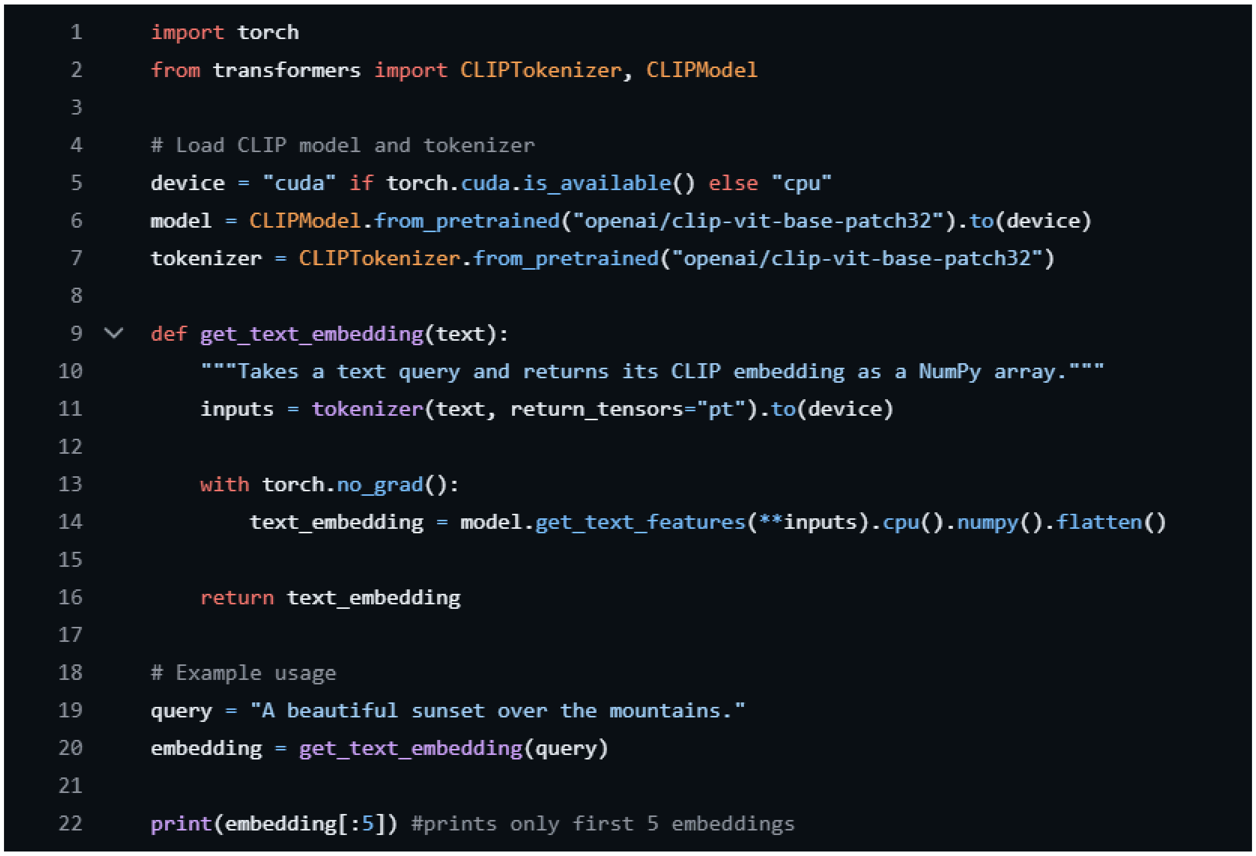

Input Query
CLIP Transform to text embedding
FAISS search for relevant image embeddings
CLIP Transform to text embedding
Polars fetch images of returned indices
Load Images in Gradio
Output Images

Fig 4.1: Architecture Diagram

4.2 Dataset and Embeddings

The system starts by loading the DSD 50,000, which is defined as a limit from the original Kaggle dataset, together with a matching CSV file having precomputed embeddings. Each CSV row contains the image embedding, text embedding, and original text description. The embeddings are subsequently loaded into memory and indexed for use.

4.3 CLIP Embedding Model



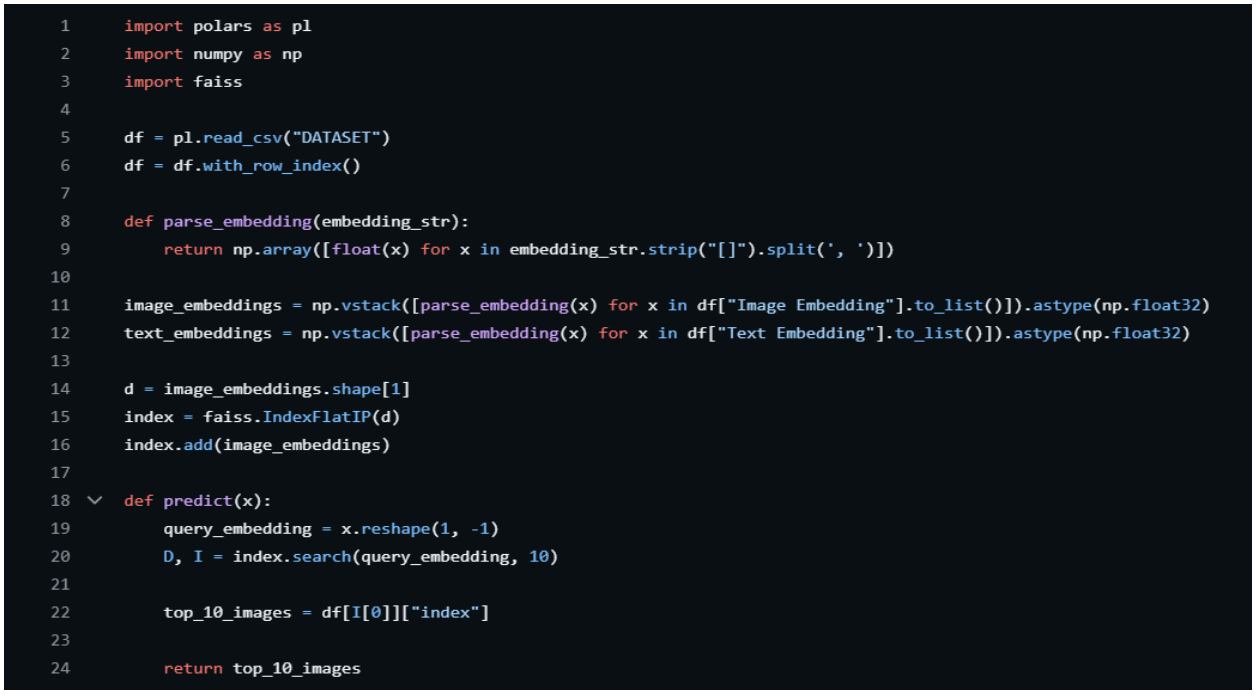
```python
import torch
from transformers import CLIPTokenizer, CLIPModel

# Load CLIP model and tokenizer
device = "cuda" if torch.cuda.is_available() else "cpu"
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32").to(device)
tokenizer = CLIPTokenizer.from_pretrained("openai/clip-vit-base-patch32")

def get_text_embedding(text):
    """Takes a text query and returns its CLIP embedding as a NumPy array."""
    inputs = tokenizer(text, return_tensors="pt").to(device)

    with torch.no_grad():
        text_embedding = model.get_text_features(**inputs).cpu().numpy().flatten()

    return text_embedding

# Example usage
query = "A beautiful sunset over the mountains."
embedding = get_text_embedding(query)

print(embedding[:5]) #prints only first 5 embeddings
```

Fig 4.2: CLIP Function to generate embeddings

CLIP is brought up in the environment and applied to embedding generation. The precomputed image embeddings are loaded onto disk for novel queries, with only the text encoder employed in that case. Separation accomplishes inference quickly and in small amounts of memory.

4.4 FAISS-Based Similarity Search

All image embeddings are then indexed with FAISS based on cosine similarity as the distance metric. When a user inputs a query, its text embedding is calculated with CLIP and fed into FAISS to get the top 10 most similar image vectors. This vector list is interpreted as corresponding image filenames for display.

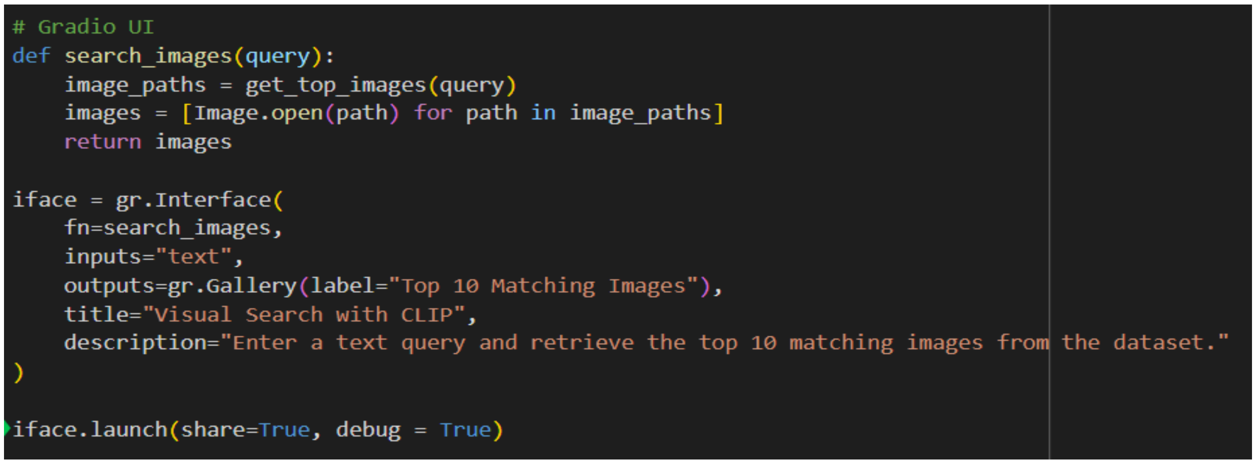


```python
import polars as pl
import numpy as np
import faiss

df = pl.read_csv("DATASET")
df = df.with_row_index()

def parse_embedding(embedding_str):
    return np.array([float(x) for x in embedding_str.strip("[]").split(', ')])

image_embeddings = np.vstack([parse_embedding(x) for x in df["Image Embedding"].to_list()]).astype(np.float32)
text_embeddings = np.vstack([parse_embedding(x) for x in df["Text Embedding"].to_list()]).astype(np.float32)

d = image_embeddings.shape[1]
index = faiss.IndexFlatIP(d)
index.add(image_embeddings)

def predict(x):
    query_embedding = x.reshape(1, -1)
    D, I = index.search(query_embedding, 10)

    top_10_images = df[I[0]]["index"]

    return top_10_images
```

Fig 4.3: Function using FAISS to retrieve top 10 image index

4.5 Gradio Interface Integration

To make it accessible and easy to use, the backend is connected to an interface created using Gradio. The interface has free-text query entry as input, processes it in real time, and shows the 10 most similar images. It also manages custom paths to the datasets, model download, and extraction process, thereby making the system completely standalone and user configurable.



```python
# Gradio UI
def search_images(query):
    image_paths = get_top_images(query)
    images = [Image.open(path) for path in image_paths]
    return images

iface = gr.Interface(
    fn=search_images,
    inputs="text",
    outputs=gr.Gallery(label="Top 10 Matching Images"),
    title="Visual Search with CLIP",
    description="Enter a text query and retrieve the top 10 matching images from the dataset."
)

iface.launch(share=True, debug = True)
```

Fig 4.4: Gradio UT Integration

RESULT AND DISCUSSIONS

5.1 Overview of System Functionality

The Vision Language Model used here CLIP successfully aligns images and their respective description, allowing us to create a system where we can search for images related to a query by comparing both their vector embeddings as generated by CLIP. Our search algorithm of choice is FAISS, which is easily parallelized on the GPU and fetches vector embeddings similar to the input given to it with a high level of accuracy.

5.2 User Interface Implementation

We use Gradio as a framework to build a UI for our visual search platform. Gradio is a popular framework to build UIs to interact with Language Models and allows us to create easy to use interfaces with minimal effort.

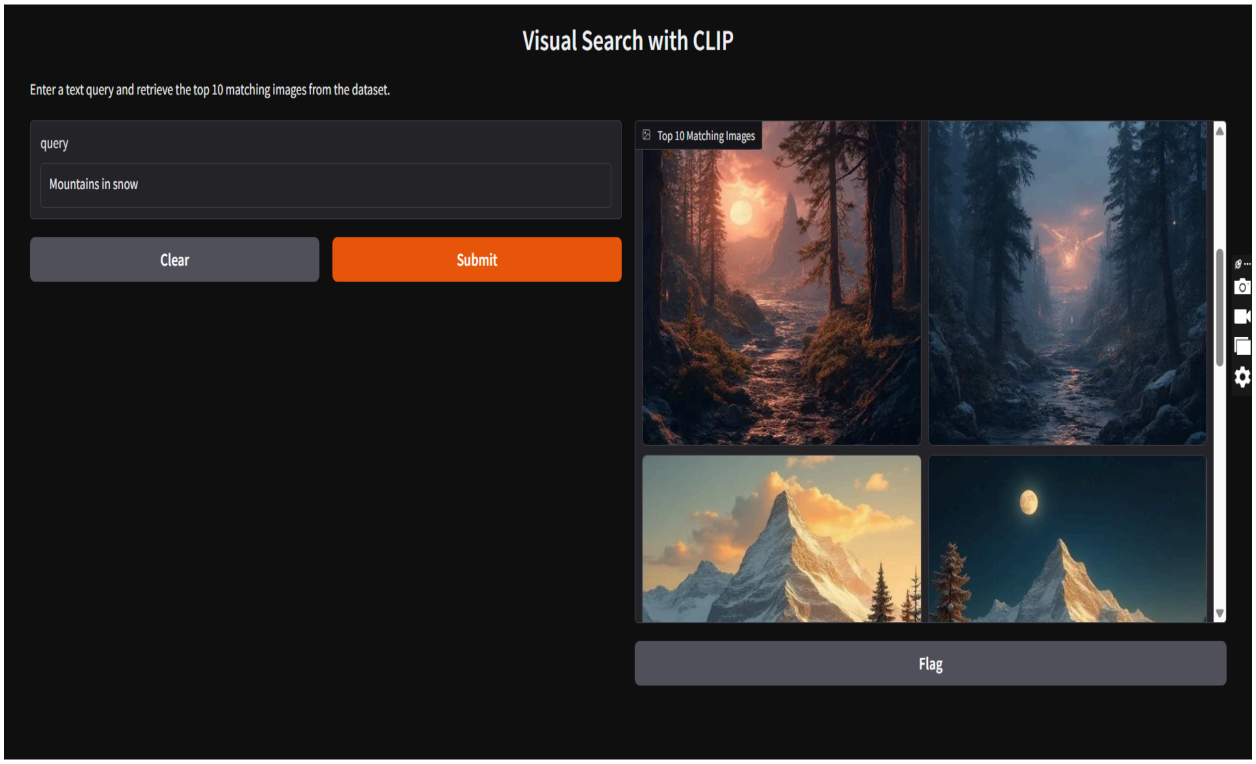


Fig 5.1: Gradio interface showcasing image retrieval results for a sample query.

5.3 Dataset Visualization

The dataset contains 50000 AI-generated images along with their description from the DSD dataset. We used this dataset to avoid having to scrape the web for images and since the descriptions given had to be related to the image since they were involved in generating the image itself.

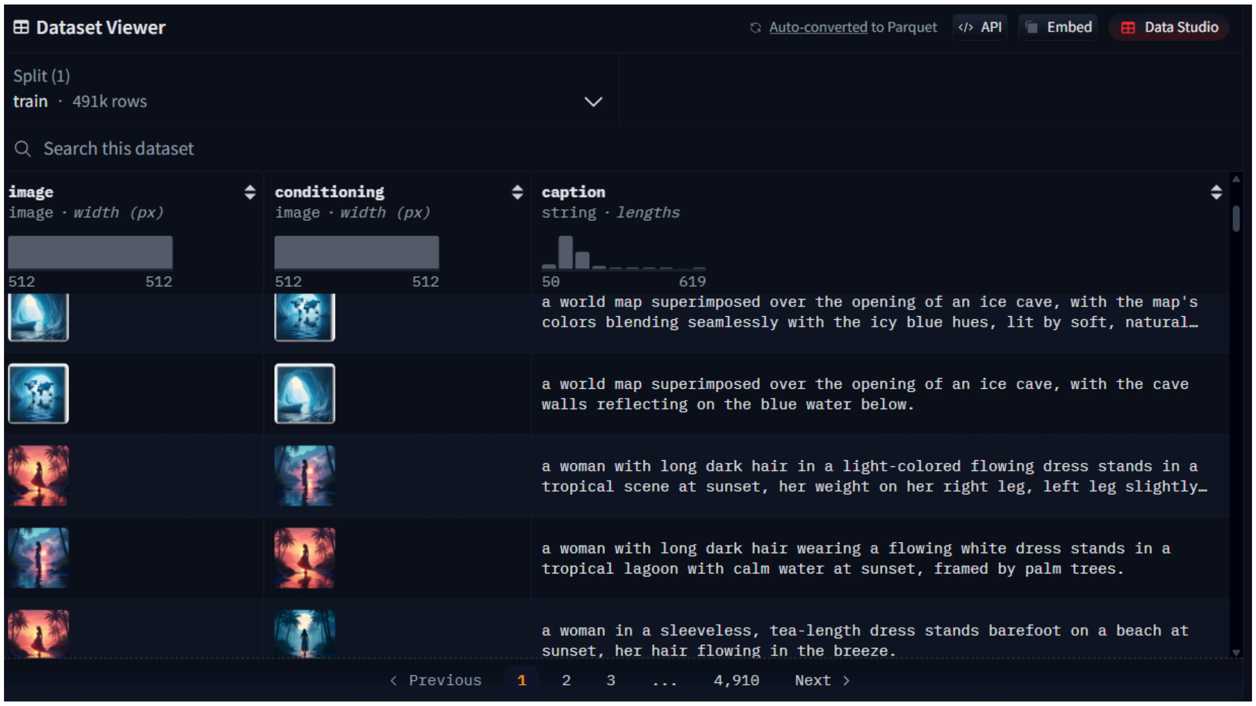
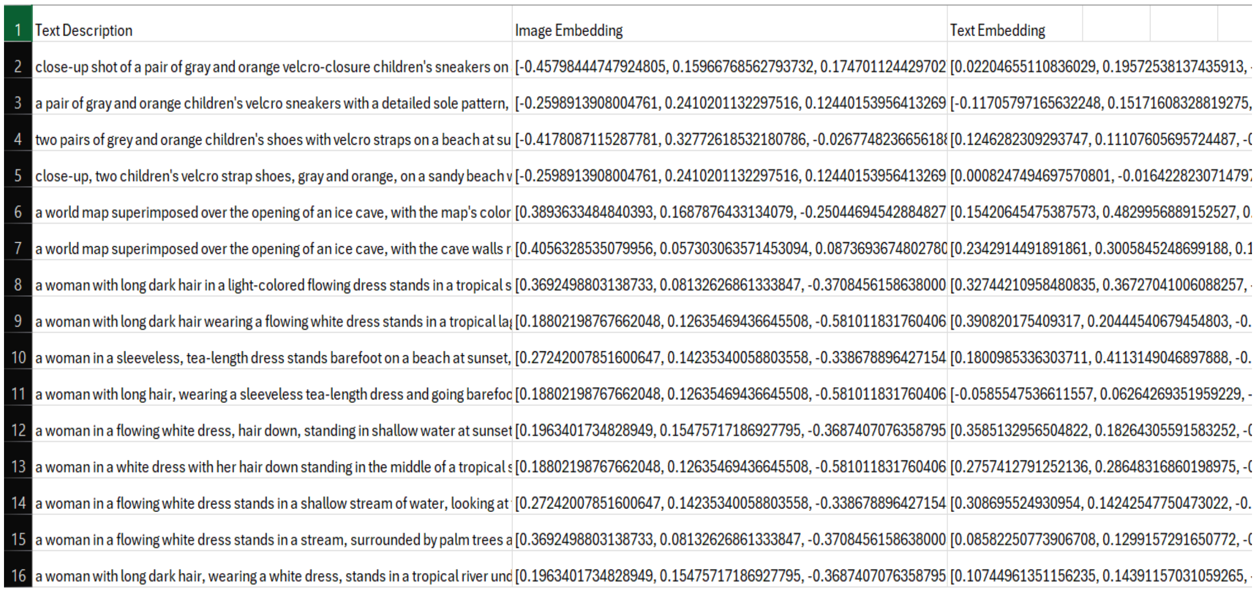


image image · width (px) 512 – 512	conditioning image · width (px) 512 – 512	caption string · lengths 50 – 619
		a world map superimposed over the opening of an ice cave, with the map's colors blending seamlessly with the icy blue hues, lit by soft, natural…
		a world map superimposed over the opening of an ice cave, with the cave walls reflecting on the blue water below.
		a woman with long dark hair in a light-colored flowing dress stands in a tropical scene at sunset, her weight on her right leg, left leg slightly…
		a woman with long dark hair wearing a flowing white dress stands in a tropical lagoon with calm water at sunset, framed by palm trees.
		a woman in a sleeveless, tea-length dress stands barefoot on a beach at sunset, her hair flowing in the breeze.

Fig 5.2: Preview of the DSD dataset available on Hugging Face.

5.4 Embedding Dataset Creation

A crucial part of the project involved precomputing and storing embeddings for both the text prompts (conditionings) and the corresponding images. These embeddings were generated using the CLIP model and stored in a CSV file, which serves as the backend data for FAISS search. A snippet from this CSV file is shown below.

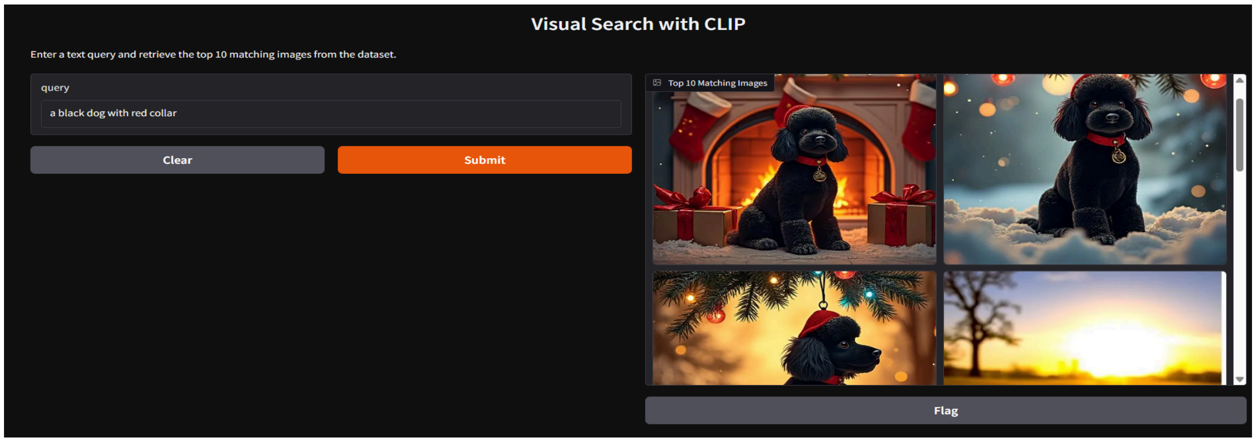


	Text Description	Image Embedding	Text Embedding			
1	Text Description	Image Embedding	Text Embedding			
2	close-up shot of a pair of gray and orange velcro-closure children's sneakers on	[-0.45798444747924805, 0.15966768562793732, 0.174701124429702	[0.02204655110836029, 0.19572538137435913,			
3	a pair of gray and orange children's velcro sneakers with a detailed sole pattern,	[-0.2598913908004761, 0.2410201132297516, 0.12440153956413269	[-0.11705797165632248, 0.15171608328819275,			
4	two pairs of grey and orange children's shoes with velcro straps on a beach at su	[-0.4178087115287781, 0.32772618532180786, -0.026774823665618	[0.1246282309293747, 0.11107605695724487, -			
5	close-up, two children's velcro strap shoes, gray and orange, on a sandy beach v	[-0.2598913908004761, 0.2410201132297516, 0.12440153956413269	[0.0008247494697570801, -0.016422823071479			
6	a world map superimposed over the opening of an ice cave, with the map's color	[0.3893633484840393, 0.1687876433134079, -0.25044694542884827	[0.15420645475387573, 0.4829956889152527, 0.			
7	a world map superimposed over the opening of an ice cave, with the cave walls r	[0.4056328535079956, 0.057303063571453094, 0.087369367480278	[0.2342914491891861, 0.3005845248699188, 0.1			
8	a woman with long dark hair in a light-colored flowing dress stands in a tropical s	[0.3692498803138733, 0.08132626861333847, -0.3708456158638000	[0.32744210958480835, 0.36727041006088257,			
9	a woman with long dark hair wearing a flowing white dress stands in a tropical la	[0.18802198767662048, 0.12635469436645508, -0.581011831760406	[0.390820175409317, 0.20444540679454803, -0.			
10	a woman in a sleeveless, tea-length dress stands barefoot on a beach at sunset,	[0.27242007851600647, 0.14235340058803558, -0.338678896427154	[0.1800985336303711, 0.4113149046897888, -0.			
11	a woman with long hair, wearing a sleeveless tea-length dress and going barefoo	[0.18802198767662048, 0.12635469436645508, -0.581011831760406	[-0.0585547536611557, 0.06264269351959229, -			
12	a woman in a flowing white dress, hair down, standing in shallow water at sunset	[0.1963401734828949, 0.15475717186927795, -0.3687407076358795	[0.3585132956504822, 0.18264305591583252, -			
13	a woman in a white dress with her hair down standing in the middle of a tropical s	[0.18802198767662048, 0.12635469436645508, -0.581011831760406	[0.2757412791252136, 0.28648316860198975, -			
14	a woman in a flowing white dress stands in a shallow stream of water, looking at	[0.27242007851600647, 0.14235340058803558, -0.338678896427154	[0.308695524930954, 0.14242547750473022, -0.			
15	a woman in a flowing white dress stands in a stream, surrounded by palm trees a	[0.3692498803138733, 0.08132626861333847, -0.3708456158638000	[0.08582250773906708, 0.1299157291650772, -			
16	a woman with long dark hair, wearing a white dress, stands in a tropical river und	[0.1963401734828949, 0.15475717186927795, -0.3687407076358795	[0.10744961351156235, 0.14391157031059265,			

Figure 5.3: Snippet of the generated CSV file containing image embeddings, text embeddings, and their textual descriptions.

5.5 Search Accuracy and Efficiency

The system demonstrates impressive search speed and accuracy, even when operating over a large dataset of 50,000 images. The FAISS index enables fast approximate nearest neighbor retrieval, ensuring responsive and relevant results. Through testing with various queries ranging from abstract concepts (e.g., "a peaceful sunset") to specific subjects (e.g., "a black dog with a red collar"), the retrieved results consistently aligned with the intended meaning of the query.



Visual Search with CLIP

Enter a text query and retrieve the top 10 matching images from the dataset.

query: a black dog with red collar

Fig 5.4: Output for a specific object query "a black dog with a red collar".

5.6 Challenges Encountered and Limitations

- **Embedding Limitations and Semantic Gaps:** The system's accuracy depends on CLIP's training data. It may struggle with abstract or culturally specific queries not well represented in the model's learned concepts.
- **High-Dimensional Data Optimization:** Working with 512-dimensional embeddings required balancing memory use and search speed. FAISS indexing helped, but optimizing without quality loss remained a challenge.

CONCLUSION

Vision Language Models (VLMs) like CLIP developed by OpenAI are very useful for building image search engines and perform very well. They result in embeddings that are easy to compare the similarities of using common algorithms and save space by only having to load the images into more active storage exactly when required after finding the most similar vector embeddings. Models like this provide us an insight into the alignment of natural language and the visual domain.

FAISS allows for a fast, lightweight and reliable algorithm for indexing and searching for similar vector embeddings, allowing us to do large scalable searches for real life image search engines which would not be extensively searchable otherwise.

These two systems along with Gradio help us build a fast and accurate visual search engine that is scalable and lightweight.