A REPORT ON VISUAL SEARCH USING VISION-LANGUAGE MODELS (VLMs)

Submitted by:-

1. SHASHANK B HUNAGUND

2. ARATHI G K

3. MAHESH KUMAR

Under the esteemed guidance of

Prof. Vinola C

Head of the Department, Internet of Things.

1.1 ABSTRACT

In the rapidly evolving world of e-commerce, traditional keyword-based search methods often fail to capture the visual essence of products that customers seek. To bridge this gap, Visual Search Using Vision-Language Models (VLMs) for Fashion Products leverages to provide an intuitive and efficient way for users to search fashion items using both images and textual descriptions. This technology enables seamless product discovery, improving user experience and customer satisfaction.

1.2 PROBLEM STATEMENT

The online fashion industry faces significant challenges in helping customers find products that match their preferences. Conventional text-based search engines rely heavily on user-provided keywords, which may not always accurately describe the desired product. Customers often struggle to find specific fashion items due to vague descriptions, lack of knowledge about product names, or language barriers. Additionally, users may come across a fashion item they like—whether from social media, magazines, or real-world interactions—but find it difficult to search for a similar product using only text-based search engines.

1.3 OBJECTIVE OF OUR PROJECT

The objective of this project is to develop an advanced visual search system that allows users to find fashion products using either textual descriptions or by uploading an image. By leveraging Vision-Language Models (VLMs), this system efficiently interprets and retrieves relevant fashion items, significantly improving search accuracy and usability. Unlike conventional search methods, which rely solely on text matching, this approach captures the deeper semantic and visual similarities between products, providing users with more precise search results.

1.4 SOLUTION APPROACH

This solution not only enhances the online shopping experience by offering a more intuitive and user-friendly search mechanism, but it also provides businesses with improved product discoverability, thereby

increasing customer engagement and boosting sales. By integrating a robust visual and text-based retrieval system, fashion retailers can cater to a broader audience, including those who prefer image-based exploration over traditional text queries. Furthermore, this system can help reduce bounce rates, as users are more likely to find what they are looking for, leading to higher conversion rates and customer satisfaction.

To address the problem statement, we have implemented a Visual Search System using Vision-Language Models (VLMs). Our approach consists of the following key steps:

1. Dataset Preparation:

- We utilize a fashion product image dataset that contains a diverse range of clothing and accessories.
- Each product is associated with relevant metadata, including descriptions and categories, to enhance retrieval accuracy.

2. Feature Extraction :

- We use Torch for efficient tensor operations and deep learning model handling.
- The CLIP (Contrastive Language-Image Pretraining) model is utilized to process both image and text inputs, ensuring effective multimodal learning.
- Image Processing: Each image is transformed using CLIP's preprocessing pipeline to generate image embeddings that capture the product's visual features.
- Text Processing: Text descriptions are tokenized and transformed into text embeddings using CLIP's language model,

ensuring that both images and text exist in a shared embedding space.

3. Indexing & Similarity Search:

- All extracted embeddings are stored in an efficient indexing structure.
- For a given query (text or image), we compute its embedding and retrieve the most relevant results using cosine similarity.

4. Search Functionality:

- Text-Based Search: Users can enter a descriptive query (e.g., "red sneakers"), and the system retrieves the closest matching fashion products.
- Image-Based Search: Users can upload an image, and the system finds visually similar fashion items.
- The search results include the top most relevant matches, displayed in an intuitive gallery format.

By implementing this AI-driven visual search system, we provide a more effective way for users to explore and purchase fashion products, ultimately transforming the online shopping experience.

1.5 Architecture Diagrams

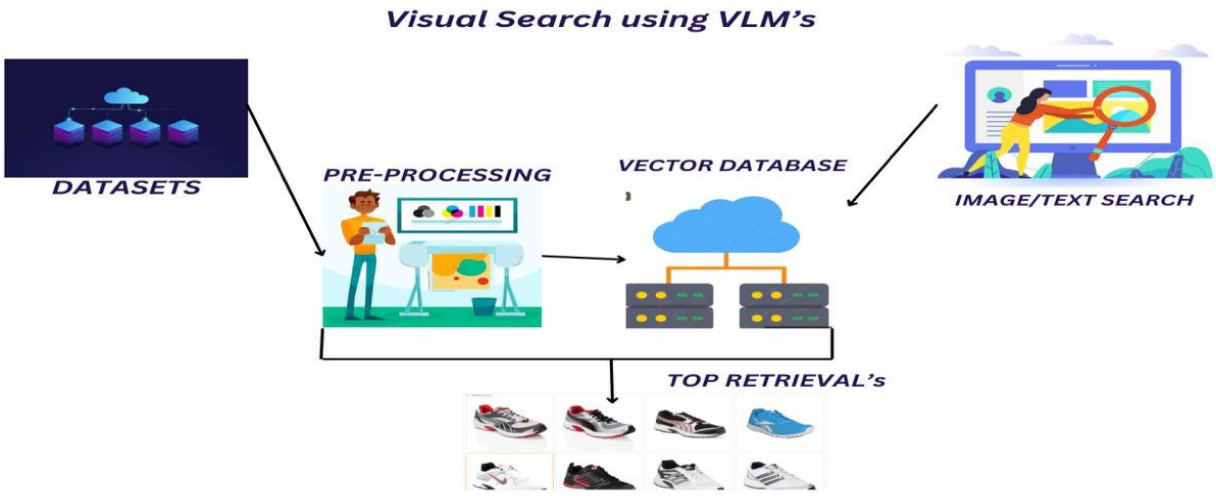

Fig 1.5.1 Architecture diagrams

1.6 SAMPLE INPUT AND OUTPUT

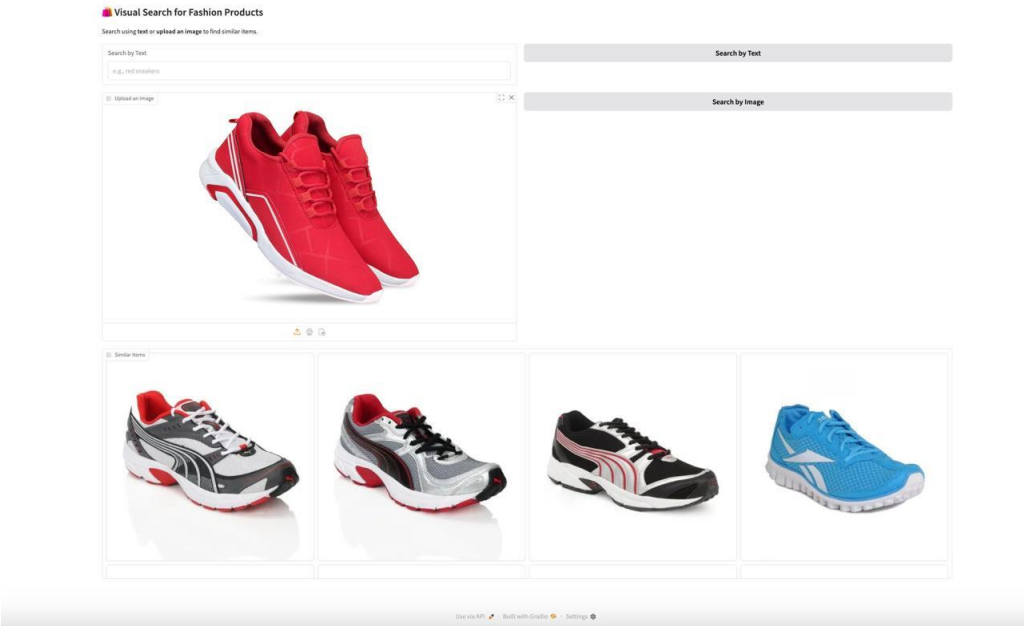

Fig 1.6.1 Search by Image

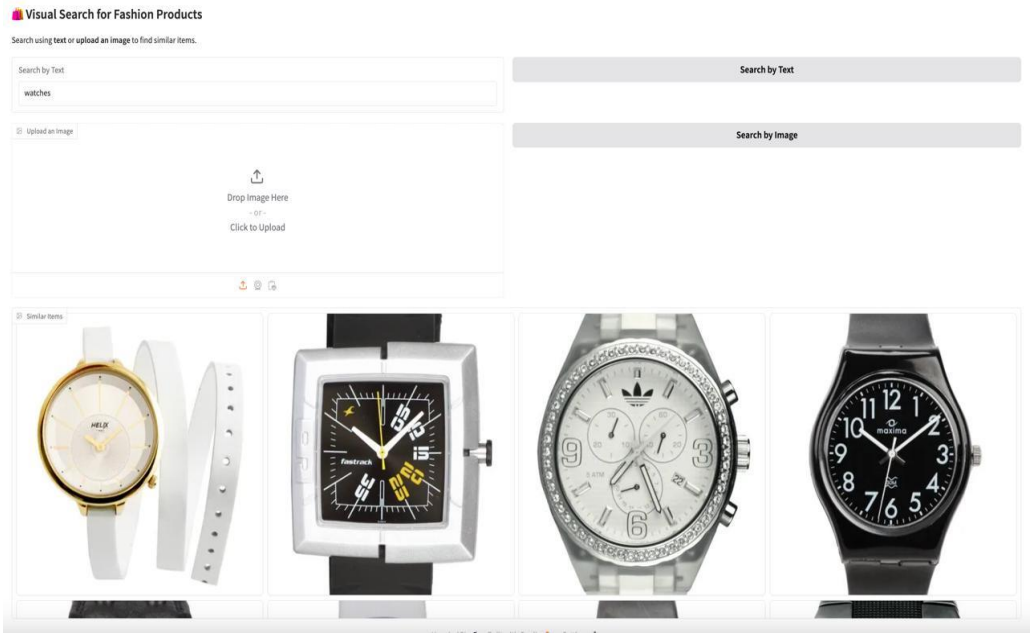

Fig 1.6.2 Search by Text

1.7 INDIVIDUAL CONTRIBUTION OF TEAM MEMBER

Person 1: [Arathi]

She played a crucial role in implementing the text-based search functionality, ensuring that users could efficiently retrieve fashion products using keywords and descriptions. They worked on optimizing search accuracy and refining the relevance of text-based queries. Additionally, she took responsibility for compiling the project report, documenting the methodologies, implementation details, and overall findings of the project in a structured manner. Their contribution ensured that the technical aspects and outcomes were well-documented for future reference.

Person 2: [Mahesh]

He collaborated with Person 1 in developing and fine-tuning the text-based search feature, focusing on improving keyword matching, search efficiency, and enhancing the overall retrieval system. They contributed to refining the backend logic to ensure seamless text-based product searches. Their expertise helped improve search response time and user experience. By working closely with the team, they ensured that the search system effectively met user requirements.

Person 3: [Shashank]

He was responsible for implementing the image-based search functionality, integrating Vision-Language Models (VLMs) to enable users to search for fashion products using images. They worked on training and optimizing the model for accurate visual recognition and matching.

1.8 CODE

```
import torch

import open_clip

import gradio as gr

from datasets import load_dataset

from torchvision import transforms

from PIL import Image

import numpy as np

# Load the dataset (fashion product images dataset)

dataset = load_dataset("ceyda/fashion-products-small",
split="train")

# Load CLIP model with correct unpacking and QuickGELU

model = open_clip.create_model("ViT-B-32-quickgelu",
pretrained="openai")

# Corrected image transform function

preprocess = open_clip.image_transform(model.visual.image_size,
is_train=False)

# Load tokenizer

tokenizer = open_clip.get_tokenizer("ViT-B-32")

# Move model to GPU if available

device = "cuda" if torch.cuda.is_available() else "cpu"

model.to(device)
```

```
# Function to compute image embeddings
def get_image_embedding(image):
    image = preprocess(image).unsqueeze(0).to(device)
    with torch.no_grad():
        image_features = model.encode_image(image)
            return image_features / image_features.norm(dim=-1,
         keepdim=True)
# Function to compute text embeddings
def get_text_embedding(text):
    text_inputs = tokenizer([text]).to(device)
    with torch.no_grad():
        text_features = model.encode_text(text_inputs)
    return text_features / text_features.norm(dim=-1,
keepdim=True)
# Precompute embeddings for all images in the dataset
image_embeddings = []
image_paths = []
for item in dataset.select(range(100)):  # Limit to 100 images for
speed
    image = item["image"]
    image_embeddings.append(get_image_embedding(image))
    image_paths.append(image)
```

```
# Stack all embeddings into a tensor
image_embeddings = torch.cat(image_embeddings, dim=0)
# Function to search for similar images based on text
def search_similar_image(query_text):
    text_embedding = get_text_embedding(query_text)
    similarities = (image_embeddings @ text_embedding.T).squeeze(1).cpu().numpy()
    # Get top 20 matches
    best_match_idxs = np.argsort(similarities)[-20:][::-1]
    return [image_paths[i] for i in best_match_idxs]
# Function to search for similar images based on an uploaded image
def search_similar_by_image(uploaded_image):
    query_embedding = get_image_embedding(uploaded_image)
    similarities = (image_embeddings @ query_embedding.T).squeeze(1).cpu().numpy()
    # Get top 20 matches
    best_match_idxs = np.argsort(similarities)[-20:][::-1]
    return [image_paths[i] for i in best_match_idxs]
# Gradio UI
with gr.Blocks() as demo:
    gr.Markdown("## □ Visual Search for Fashion Products")
```

```
    gr.Markdown("Search using *text* or *upload an image* to find similar items.")

    with gr.Row():

        query_input = gr.Textbox(label="Search by Text", placeholder="e.g., red sneakers")

    search_button = gr.Button("Search by Text")

    with gr.Row():

        image_input = gr.Image(type="pil", label="Upload an Image")

        image_search_button = gr.Button("Search by Image")

        output_gallery = gr.Gallery(label="Similar Items", columns=4, height=500)

    search_button.click(search_similar_image, inputs=[query_input], outputs=[output_gallery])

    image_search_button.click(search_similar_by_image, inputs=[image_input], outputs=[output_gallery])

    demo.launch(share=True)
```

1.9 CODE EXPLANATION

☆ Imports and Dependencies

The implementation begins by importing essential Python Libraries:

● torch: Used for tensor operations and computations on GPU

● open_clip: Provides access to the CLIP model for feature Extraction:

● gradio: Used for building the interactive user interface.

● datasets: Used to load a dataset containing fashion product Images:

● torchvision.transforms: Includes image preprocessing utilities.

● PIL (Python Imaging Library): Handles image processing.

● numpy: Used for numerical computations.

☆ Dataset Loading:

The fashion product dataset is loaded using the load_dataset function, which retrieves product images for training and testing.

☆ CLIP Model Initialization:

● The CLIP model is loaded using open_clip.create_model(), which allows encoding both images and text into a shared feature space.

● The image preprocessing function open_clip.image_transform() ensures that input images are transformed to match the model's required format.

● The tokenizer for text processing is obtained using open_clip.get_tokenizer().

● The model is moved to GPU (if available) using model.to(device) for faster processing

☆ Feature Extraction:

●The function get_image_embedding(image) extracts image embeddings by passing the image through the CLIP model.

● The function get_text_embedding(text) generates text embeddings using the CLIP tokenizer and text encoder.

☆ Precomputing Image Embeddings:

The embeddings for 100 images from the dataset are precomputed and stored in a tensor to speed up the search process.

☆ Similarity Search:

● Text-based search (search_similar_image): Computes the text query embedding and finds the most similar images using cosine similarity.

● Image-based search (search_similar_by_image): Computes the uploaded image's embedding and finds the closest matching images in the dataset.

☆ User Interface with Gradio:

● Gradio Blocks UI provides an easy-to-use interface where users can:

1. Search by text (inputting a description like “red sneakers”).

2. Search by image (uploading a product image to find similar items).

● Search results are displayed as an image gallery, showing the top matches.

1.10 LOGIC OF CODE:

1. Data Preparation & Model Loading:

● The system first loads the dataset and initializes the CLIP model.

● It preprocesses the dataset’s images and computes embeddings in advance for efficient retrieval.

2. Feature Extraction & Indexing:

● Every product image is converted into a high-dimensional embedding.

● Text queries are also transformed into embeddings, ensuring they exist in the same feature space as the images.

3. Similarity Computation:

● When a user searches for a product (via text or image), the system computes an embedding for the query.

- The query embedding is compared against the precomputed embeddings using cosine similarity.
- The top most similar items are retrieved and displayed in a gallery.

4. User Interaction & Display:

- Users can search using keywords or upload an image.
- The system instantly retrieves and displays matching fashion items.

This approach ensures efficient, real-time search results, enhancing the online shopping experience for users and improving product discoverability for fashion retailers.