Comprehensive Report: T-shirt Image Retrieval and Captioning System

Mahla Entezari

January 2025

Abstract

This report details the design, implementation, and evaluation of a comprehensive T-shirt image retrieval and captioning system. Leveraging state-of-the-art pre-trained deep learning models from the Hugging Face ecosystem, the system addresses key challenges in multimodal information retrieval. It encompasses three core modules: text-to-image retrieval using CLIP and FAISS for efficient similarity search; automated image captioning using BLIP for generating descriptive textual metadata; and text-to-text retrieval based on these generated captions. The project emphasizes the practical application of pre-trained models to build robust AI solutions without the need for extensive training from scratch. The report provides a thorough explanation of the methodology, model justifications, system implementation, and a qualitative assessment of the results, concluding with limitations and future directions for enhancement and scalability.

0. Contents

1. Introduction to Multimodal Information Retrieval

The rapidly expanding volume of digital content necessitates advanced methods for efficient information retrieval. While traditional retrieval systems often rely on single modalities, such as text-based search for documents or metadata-driven search for images, the true value of modern data often lies in its **multimodal nature**. Images are not merely visual data; they often come with associated text (captions, descriptions, tags), or their content can be best understood through textual interpretation. This project delves into the exciting field of multimodal information retrieval, specifically focusing on building a robust system for T-shirt images. The primary objective is to enable users to search for T-shirts using natural language queries and to automatically generate descriptive text for these visual assets.

The increasing availability of large, pre-trained deep learning models has revolutionized the development of such systems. These models, often trained on vast and diverse datasets, have learned powerful representations of both visual and textual information, capable of understanding complex semantic relationships. Leveraging these pre-trained models eliminates the need for extensive data collection and model training from scratch, significantly reducing development time and computational resources. This project strictly adheres to the principle of utilizing pre-trained models from the Hugging Face ecosystem, showcasing their immediate applicability to real-world tasks.

The system is compartmentalized into three interconnected modules:

- **Image Retrieval (Text-to-Image)**: This module allows users to provide a textual query (e.g., "blue graphic T-shirt") and receive the most semantically similar T-shirt images from a dataset of 1,993 images. This task explores contrastive learning principles and the direct application of pre-trained vision-language models for cross-modal similarity.
- **Image Captioning**: This module extends the system's capabilities by generating detailed textual descriptions for each T-shirt image. These machine-generated captions serve as valuable metadata, enriching the images with linguistic context.
- **Text Retrieval (Text-to-Text via Captions)**: Building upon the generated captions, this module enables a more refined text-based search. Given a textual query, the system retrieves images by identifying captions that are most relevant to the query. This involves leveraging text embedding models and similarity computation techniques.

This report will systematically detail the methodology, implementation, and results for each of these subprojects. It will include comprehensive documentation, justification for the chosen pre-trained models, and conceptual standalone inference pipelines to demonstrate the system's capabilities. The overarching goal is to provide a practical understanding of multimodal IR and the immense potential of pre-trained models in bridging the gap between language and vision.

2. Dataset Description: T-shirt Image Collection

2.1. Overview and Source

The dataset central to this project comprises a collection of T-shirt images, specifically provided for hands-on experience in information retrieval. The dataset's focused nature on a single product category (**T-shirts**) allows for a deep dive into visual and textual characteristics pertinent to e-commerce or fashion applications.

2.2. Data Volume and Composition

The dataset consists of **1,993 distinct T-shirt images**. Each image is a visual representation of a T-shirt, potentially featuring various colors, patterns, designs, graphics, and styles. The images are typically in standard image formats such as JPG, PNG, or JPEG.

2.3. Key Characteristics Relevant to Information Retrieval

- **Homogeneous Product Category**: All images belong to the same product category (T-shirts). This homogeneity can simplify some aspects of modeling (e.g., general object recognition is not the primary challenge) but highlights the need for models to distinguish subtle variations within the category (e.g., different types of graphics, shades of color, fabric textures).

- **Absence of Explicit Metadata**: Critically, the provided dataset of 1,993 images does not come with pre-existing textual metadata such as product descriptions, tags, or manual captions. This absence directly motivates the image captioning task, where the system must generate this information automatically, and the text retrieval task, which then leverages these generated captions.
- **Variety in Visual Attributes**: Despite being a single product category, T-shirts can exhibit a wide array of visual attributes:
  - **Colors**: From basic black and white to vibrant hues, pastels, and multi-color combinations.
  - **Patterns/Graphics**: Solid, striped, polka-dotted, abstract art, band logos, cartoon characters, text-based graphics, etc.
  - **Style**: Crew neck, V-neck, long sleeve, short sleeve, sleeveless, fitted, oversized, etc.
  - **Texture**: Cotton, polyester, blended fabrics (though visual distinction might be subtle).
- **Image Quality and Consistency**: While the exact characteristics vary, a typical dataset of this nature would exhibit reasonable image quality, though consistency in background, lighting, and T-shirt presentation (e.g., on a model, flat lay) might fluctuate. Preprocessing steps are essential to standardize inputs for deep learning models.
- **Scalability Consideration**: While 1,993 images is a manageable size for initial experimentation, any practical image retrieval system needs to consider scalability to millions or billions of items. The choice of embedding models and indexing strategies (like FAISS) is crucial for this.

The clean nature of the image files, without immediate concerns of corruption or highly irregular formats, provides a solid foundation for directly applying state-of-the-art computer vision and natural language processing techniques. The challenge lies in extracting meaningful, searchable features and descriptions from this purely visual input.

3. Methodology

My methodology for developing this multimodal information retrieval system for T-shirt images is structured into distinct, yet interconnected, phases. Each phase leverages pre-trained models from Hugging Face, adhering to the project's core constraint of avoiding model training from scratch.

3.1. Data Loading and Preparation

- **Image Collection**: The first step involves gathering all 1,993 T-shirt image files from the designated dataset path. The file names are extracted to serve as unique identifiers and pointers to the image data.
- **Image Preprocessing for Models**: Different pre-trained models often have specific input requirements (e.g., image resolution, normalization parameters).
  - For CLIP (Image Encoder), images are resized to **224x224 pixels** and normalized using a specific mean and standard deviation derived from its training data. This ensures consistency with the model's learned representations.
  - For BLIP (Image Captioning), the model's own processor handles the internal resizing and normalization, abstracting away these details for the user.

  Images are consistently converted to **RGB format** to ensure uniformity.

3.2. Image Retrieval (Text-to-Image)

This task focuses on retrieving visually similar T-shirts given a textual query.

3.2.1. Feature Extraction (Image Embeddings)

- **Model Selection**: The `openai/clip-vit-base-patch32` model is chosen for its ability to learn a joint embedding space for images and text.
- **Process**: For each of the 1,993 T-shirt images, the preprocessed image is fed into CLIP's image encoder. The output is a fixed-size, high-dimensional vector (e.g., **512-dimensional** for base-patch32) that semantically represents the image. This process is performed offline and once to create a searchable database of image features.
- **Storage**: These image embeddings, along with their corresponding image file paths, are stored in memory or persisted to disk (e.g., as NumPy arrays or JSON files).

3.2.2. Indexing for Efficient Search (FAISS)

To enable rapid nearest-neighbor search over the collection of image embeddings, a **FAISS** (Facebook AI Similarity Search) index is employed.

- **Index Type**: A `faiss.IndexFlatL2` index is initialized. This index performs a brute-force L2 (Euclidean) distance search, guaranteeing exact nearest neighbors. For larger datasets, approximate nearest neighbor (ANN) indices would be considered.
- **Adding Embeddings**: The pre-computed image embeddings are added to the FAISS index, making them searchable. This indexing process is also performed offline.

3.2.3. Query Processing and Retrieval

- **Text Embedding**: When a user inputs a textual query (e.g., "blue graphic T-shirt"), CLIP's text encoder is used to transform this query into a corresponding fixed-size text embedding within the same joint embedding space.
- **Similarity Search**: The query's text embedding is then used to query the FAISS index. FAISS returns the `top_k` most similar image embeddings (based on L2 distance) and their original indices in the dataset.
- **Result Presentation**: The images corresponding to these `top_k` indices are retrieved from their file paths and displayed to the user, providing a visual representation of the retrieval results.

3.3. Image Captioning

This task involves generating descriptive textual captions for the T-shirt images.

3.3.1. Captioning Model Selection

- **Model**: The `Salesforce/blip-image-captioning-base` model is chosen. BLIP is a powerful Vision-Language Model capable of generating coherent and contextually relevant captions.

3.3.2. Caption Generation Process

- For each T-shirt image, the image is passed through the BLIP model's processor and then its conditional generation component.
- The model outputs a sequence of tokens, which are then decoded back into a human-readable string, forming the image caption.
- This process is typically performed offline for all images to pre-compute captions.
- **Caption Storage**: The generated captions are stored in a structured format (e.g., a dictionary mapping image paths to captions, saved as a JSON file) for easy access in subsequent tasks.

3.4. Text Retrieval (Text-to-Text via Captions)

This task enhances retrieval by leveraging the generated captions.

3.4.1. Text Embedding (Captions and Query)

- **Model Reuse**: CLIP (`openai/clip-vit-base-patch32`) is reused for its text encoding capabilities. This is advantageous because it embeds both the query and the captions into the same semantic space, allowing for direct comparison.
- **Process**:
  - The user's textual query is converted into a query embedding using CLIP's text encoder.
  - All pre-generated captions for the T-shirt images are also converted into individual caption embeddings using the same CLIP text encoder. This is a one-time offline process if not done already for the captions.

3.4.2. Similarity Computation

- **Metric**: **Cosine similarity** is used to measure the semantic resemblance between the query embedding and each of the caption embeddings. Cosine similarity is preferred here as it captures the angular relationship between vectors, making it robust to vector magnitude differences and ideal for semantic comparison.
- **Ranking**: The captions (and thus their corresponding images) are ranked in descending order based on their cosine similarity scores with the query embedding.

3.4.3. Result Presentation

The `top_k` images with the highest-ranked captions are retrieved and displayed. Alongside each image, its generated caption is shown, providing immediate context for why that image was retrieved.

This comprehensive methodology ensures that the T-shirt image retrieval and captioning system is robust, efficient, and leverages the full potential of pre-trained multimodal deep learning models.

4. Preprocessing and Embedding Strategies

Effective preprocessing and embedding generation are foundational to the success of any deep learning-based information retrieval system. This section elaborates on the specific strategies employed for both image and text modalities.

4.1. Image Preprocessing

For both CLIP (Image Encoder) and BLIP (Image Captioning), consistent image preprocessing is crucial to align the input data with the models' expectations during their pre-training phase.

4.1.1. Image Loading and Format Conversion

- All images are loaded using the **PIL (Pillow) library**.
- They are immediately converted to **RGB format** (`.convert("RGB")`). This is a standard practice to ensure a consistent three-channel input, as many deep learning models are trained on RGB images. This prevents issues that might arise from grayscale (single-channel) or other image formats.

4.1.2. Resizing

- Deep learning models, particularly Vision Transformers like CLIP's image encoder, expect a fixed input resolution. For `openai/clip-vit-base-patch32`, this resolution is **224x224 pixels**.
- `torchvision.transforms.Resize((224, 224))` is applied. This rescales the shorter side of the image to 224 pixels and then crops the center, or rescales both dimensions independently to 224 pixels, depending on the specific implementation detail. The goal is to fit the image into the model's required input dimensions while maintaining aspect ratio as much as possible or centering content.

4.1.3. Tensor Conversion

- `torchvision.transforms.ToTensor()` converts the `PIL.Image` into a **PyTorch Tensor**.
- This transformation also automatically scales pixel values from the `[0, 255]` integer range to the `[0.0, 1.0]` floating-point range. This normalization is a common requirement for neural networks, as it helps in stabilizing gradients during training (though here we are only inferring).

4.1.4. Normalization (CLIP Specific)

- `torchvision.transforms.Normalize(mean=[0.481, 0.457, 0.408], std=[0.268, 0.261, 0.275])` is a critical step specific to CLIP's image encoder.
- **Rationale**: CLIP was pre-trained on a massive dataset of images (like ImageNet, or similar large-scale datasets) with these specific mean and standard deviation values applied to each color channel (R, G, B). For the model to interpret visual features correctly, the input images during inference must undergo the exact same normalization as during its training.
- **Impact of Incorrect Normalization**: If different mean and standard deviation values are used:
  - The input distribution will shift, causing the model to receive data outside the numerical range it was trained on.
  - This can lead to inaccurate feature activations within the model's layers.
  - Consequently, the generated image embeddings will not align correctly in the CLIP's joint embedding space, leading to distorted cosine similarity calculations and poor retrieval performance.
- **BLIP's Internal Handling**: For BLIP, the `BlipProcessor` handles these normalization steps internally. When you pass a PIL image to `blip_processor(images=image, return_tensors="pt")`, it performs the necessary resizing, tensor conversion, and normalization according to the BLIP model's pre-training specifications without explicit user intervention. This simplifies the pipeline for BLIP.

4.2. Text Preprocessing and Tokenization

For both CLIP (Text Encoder) and any text-based components of BLIP, text preprocessing mainly involves tokenization and input formatting.

4.2.1. Tokenization

- Hugging Face's `transformers` library provides `CLIPProcessor` (for CLIP) and `BlipProcessor` (for BLIP), which wrap model-specific tokenizers (e.g., Byte-Pair Encoding or WordPiece).
- **Process**: When a text query or image caption is passed to `processor(text=[your_text], return_tensors="pt", padding=True, truncation=True)`, the following occurs:
  - The text is tokenized into a sequence of subword units.
  - These tokens are mapped to their corresponding numerical IDs from the model's vocabulary.
  - Special tokens (e.g., `[CLS]` for classification/embedding, `[SEP]` for separation, `[PAD]` for padding) are added.
- **Padding**: `padding=True` ensures that all tokenized sequences in a batch are padded to the length of the longest sequence, or to a maximum length if `max_length` is specified. This creates uniform-sized tensors required by neural networks.
- **Truncation**: `truncation=True` truncates sequences that exceed the model's maximum input length (e.g., **77 tokens** for CLIP).
- **Attention Mask**: An `attention_mask` is generated alongside the input IDs. This mask tells the model which tokens are actual content and which are padding, ensuring that padding tokens do not contribute to attention calculations.

4.3. Embedding Generation

Once preprocessed, both images and text are converted into dense vector embeddings.

4.3.1. Image Embeddings (CLIP)

- The preprocessed image tensors (`pixel_values`) are passed to `clip_model.get_image_features(**inputs)`.
- CLIP's image encoder (a Vision Transformer) processes these visual inputs and outputs a fixed-size embedding (e.g., a **512-dimensional** `torch.Tensor`).
- These embeddings represent the semantic content of the images in the joint image-text embedding space. They are typically normalized to unit vectors, which is beneficial for cosine similarity calculations.

4.3.2. Text Embeddings (CLIP)

- The tokenized text inputs (including `input_ids` and `attention_mask`) are passed to `clip_model.get_text_features`
- CLIP's text encoder (a Transformer-based text model) processes these linguistic inputs and outputs a fixed-size embedding (e.g., a **512-dimensional** `torch.Tensor`).
- These embeddings represent the semantic content of the text in the same joint image-text embedding space. They are also typically normalized to unit vectors.

4.3.3. Caption Generation (BLIP)

- For BLIP, `blip_model.generate(**inputs)` performs a different type of "embedding" in the sense that it generates a sequence of tokens, not a single fixed-size embedding.
- Internally, BLIP uses its image encoder to create an image representation, and its text decoder then uses this representation to generate the caption text. The output is a tensor of token IDs, which are then decoded back into a human-readable string.

By meticulously following these preprocessing and embedding strategies, the system ensures that the input data is correctly formatted and aligned with the sophisticated pre-trained models, maximizing their performance in the T-shirt image retrieval and captioning tasks.

5. System Implementation and Components

The system's implementation is modular, with distinct components for each task, facilitating clarity, maintenance, and potential future expansion. All components are built upon PyTorch and Hugging Face's `transformers` library.

5.1. Core Libraries and Environment

The project relies on a standard Python data science stack:

- `torch` and `torchvision`: Fundamental for deep learning operations, especially tensor manipulation, model inference, and image transformations.
- `transformers`: The core library for loading and interacting with pre-trained CLIP and BLIP models and their processors.
- `faiss-cpu`: Essential for efficient similarity search over large collections of high-dimensional embeddings. The `cpu` version is used for broader compatibility, though a `gpu` version (`faiss-gpu`) exists for faster computations if a CUDA-enabled GPU is available.
- `numpy`: For numerical operations, particularly for handling embeddings as arrays.
- `PIL (Pillow)`: For opening, manipulating, and saving image files.
- `matplotlib`: For visualizing images and plotting results (e.g., retrieved images).

- sklearn.metrics.pairwise.cosine_similarity: Used for calculating semantic similarity between text embeddings, especially in the Text Retrieval task.
- os and json: For file system interactions (listing image files, constructing paths) and saving/loading structured data like captions or embedding paths.

5.2. Image Retrieval Module Implementation

5.2.1. Offline Embedding Generation and Indexing

This is a crucial pre-computation step to prepare the image database for fast queries.

- preprocess_image_for_clip(img_path) Function:
  - Takes the path to an image file.
  - Loads the image using PIL.Image.open() and converts it to RGB.
  - Applies the torchvision.transforms.Compose pipeline with Resize((224, 224)), ToTensor(), and Normalize(mean=[0.481, 0.457, 0.408], std=[0.268, 0.261, 0.275]). This ensures that every image is transformed into a PyTorch tensor ready for CLIP's image encoder, adhering strictly to the model's pre-training normalization.
  - unsqueeze(0) is used to add a batch dimension, as models expect inputs in a batch format.
- Embedding Loop:
  - A loop iterates through all image files in the dataset_path.
  - For each image, preprocess_image_for_clip is called.
  - clip_model.get_image_features(**inputs) is invoked within a torch.no_grad() context. This extracts the **512-dimensional** embedding for the image. torch.no_grad() is vital as it deactivates gradient computation, reducing memory consumption and speeding up inference.
  - The embeddings are stored as NumPy arrays (.cpu().numpy()) and collected in img_embeddings_list, while corresponding paths are stored in img_paths_list.
- FAISS Index Creation:
  - img_embeddings_np = np.vstack(img_embeddings_list) concatenates all embeddings into a single NumPy array, where each row is an image embedding.
  - index = faiss.IndexFlatL2(img_embeddings_np.shape[1]) initializes a FAISS index. IndexFlatL2 is chosen for its simplicity and guarantees exact nearest neighbors by computing L2 (Euclidean) distance. img_embeddings_np.shape[1] provides the dimensionality of the embeddings (e.g., 512).
  - index.add(img_embeddings_np) adds all generated embeddings to the FAISS index.
- Persistence: The index and image paths are optionally saved to disk (faiss.write_index, json.dump) to avoid re-computing them in future runs.

5.2.2. Online Retrieval

This function handles live queries after the index is prepared.

- retrieve_images_from_query(text_query, top_k=10) Function:
  - Takes a text_query string and the desired number of results top_k.
  - Uses clip_processor(text=[text_query], return_tensors="pt") to tokenize and prepare the query.
  - clip_model.get_text_features(**inputs) generates the **512-dimensional** embedding for the textual query.
  - index.search(text_embedding, top_k) performs the rapid search in the FAISS index. It returns distances (L2 distances) and indices (array of indices of the top_k results).
  - Results are visualized using matplotlib.pyplot, opening the corresponding images from img_paths_list and displaying them.

5.3. Image Captioning Module Implementation

This module focuses on generating textual descriptions for individual images.

- `generate_caption(image_path)` Function:
  - Takes the path to an image.
  - Loads the raw image using `PIL.Image.open()` and converts it to RGB.
  - Uses `blip_processor(images=image, return_tensors="pt")` to preprocess the image. This internal processing handles resizing, normalization, and tensor conversion specific to the BLIP model.
  - `blip_model.generate(**inputs)` performs the core caption generation. This method internally manages attention mechanisms and decoding strategies (like beam search) to produce a coherent textual output.
  - `blip_processor.decode(output[0], skip_special_tokens=True)` converts the generated token IDs back into a human-readable string, removing any padding or special tokens.
- Batch Caption Generation (Helper): `generate_captions_for_all_images(image_paths_list)` is a utility to iterate through all images and generate a caption for each, typically performed as a one-time pre-computation. The results would be stored in a dictionary mapping image paths to captions.

5.4. Text Retrieval Module Implementation

This module leverages the generated captions for text-to-text retrieval.

- `retrieve_images_by_text(query, image_paths, captions, top_k=5)` Function:
  - Takes the query string, the list of all `image_paths`, and the list of all `captions` (which are assumed to be pre-generated and ordered consistently with `image_paths`).
  - **Query Embedding**: Uses `clip_processor` and `clip_model.get_text_features()` to generate the embedding for the input query. This query embedding is reshaped to `(1, -1)` to match the expected input shape for `cosine_similarity`.
  - **Caption Embeddings**: Iterates through all captions. For each caption, `clip_processor` and `clip_model.get_text_features()` are used to generate a text embedding. These are collected into `caption_embeddings_list`.
  - **Similarity Calculation**: `cosine_similarity(query_embeddings, caption_embeddings_np)` is invoked. This function from `sklearn.metrics.pairwise` computes the cosine similarity between the query embedding and every caption embedding. Cosine similarity is chosen for its effectiveness in capturing semantic closeness of text vectors.
  - **Ranking**: `np.argsort(similarities[0])[::-1]` sorts the resulting similarity scores in descending order, returning the indices of the most relevant captions.
  - **Result Extraction**: The top `top_k` image paths and their corresponding captions are retrieved using these ranked indices.
- `display_retrieved_images_with_captions(query_text_input, image_paths_list, captions_list, top_k=5)` Function: Orchestrates the text retrieval and then displays the retrieved images along with their captions for visual inspection.

This detailed breakdown demonstrates the functional implementation of each system component, highlighting the specific roles of chosen pre-trained models and auxiliary libraries in achieving the desired information retrieval and captioning capabilities.

6. Pre-trained Model Justification: A Detailed Review

The selection of pre-trained models is a cornerstone of this project, aligning with the objective of building advanced AI systems without resource-intensive training from scratch. This section provides an in-depth justification for each chosen model, highlighting their strengths and suitability for the specific tasks.

6.1. CLIP (`openai/clip-vit-base-patch32`) for Image Retrieval (Text-to-Image)

6.1.1. Model Description

CLIP (Contrastive Language-Image Pre-training) is a pioneering multimodal model developed by OpenAI. It consists of two separate encoders: a **Vision Transformer (ViT)** for images and a Transformer-based text encoder. CLIP is trained on an enormous dataset of **400 million (image, text) pairs** collected from the internet (e.g., AltText dataset). The training objective is contrastive: to maximize the cosine similarity between the embeddings of paired images and texts, while minimizing it for unpaired ones. This results in a shared, high-dimensional embedding space where semantic meaning is captured across modalities. The `base-patch32` variant refers to a specific architecture of the ViT image encoder (base size, with 32x32 pixel patches).

6.1.2. Justification for Image Retrieval (Text-to-Image)

- **Cross-Modal Understanding**: CLIP's fundamental strength lies in its ability to bridge the semantic gap between text and images. By embedding both modalities into a common space, it inherently understands which textual descriptions correspond to which visual concepts. This "cross-modal understanding" is precisely what is needed for text-to-image retrieval. A textual query can be directly compared (via cosine similarity) to an image embedding to find the most relevant visual content.
- **Zero-Shot Capability**: One of CLIP's most remarkable features is its zero-shot transferability. It can perform well on tasks involving concepts it has never explicitly seen during training, simply by understanding the relationship between new images and their corresponding text descriptions. This makes it incredibly powerful for a fixed dataset of T-shirt images where new, diverse queries might be introduced.
- **Semantic Coherence**: The embeddings produced by CLIP are highly semantic. This means that vector distances (or similarities) in the embedding space directly correspond to semantic closeness. "Blue graphic T-shirt" will be close to embeddings of actual blue graphic T-shirts, even if the model has never seen this exact combination during training or in the T-shirt dataset.
- **Computational Efficiency for Retrieval**: Once image embeddings are pre-computed and indexed (e.g., with FAISS), retrieval is extremely fast. The time-consuming part is only done once, making the system scalable for real-time queries.
- **Robustness**: Trained on an unprecedented scale of diverse internet data, CLIP has learned a robust and generalizable understanding of a vast array of visual and linguistic concepts, making it well-suited to handle the variability in T-shirt designs.

6.2. BLIP (`Salesforce/blip-image-captioning-base`) for Image Captioning

6.2.1. Model Description

BLIP (Bootstrapping Language-Image Pre-training) is a more recent and advanced Vision-Language Model developed by Salesforce Research. It builds upon earlier VLM architectures and introduces innovative techniques for effectively learning from noisy web data. BLIP integrates an image encoder (like a ViT), a text encoder, and a text decoder within a unified framework. Its key innovation is a "Med-VQA" (Multimodal Mixture of Encoder-Decoder) architecture and a "Captioning and Filtering" (CapFilt) mechanism that jointly learns image-text alignment and purifies noisy captions in web datasets, leading to improved performance across various VLM tasks, including image captioning. The base variant refers to its model size.

6.2.2. Justification for Image Captioning

- **Generative Power**: Unlike CLIP, which is primarily designed for alignment and retrieval, BLIP incorporates a text decoder specifically trained for language generation. This makes it directly capable of producing coherent, fluent, and semantically rich textual descriptions of images, fulfilling the core requirement of the image captioning task.

- **State-of-the-Art Performance**: BLIP consistently achieves high scores on public Vision-Language Model leaderboards (such as the Open VLM Leaderboard) for image captioning benchmarks (e.g., MS COCO). This indicates its ability to produce high-quality captions that accurately reflect image content.
- **Robustness to Diverse Imagery**: Its unique training methodology, which involves learning from and filtering noisy web data, makes BLIP highly robust to various image styles and contents. This is beneficial for a T-shirt dataset that may contain diverse designs and photographic conditions.
- **Unified Multimodal Understanding**: BLIP's internal architecture allows it to understand visual features in the context of generating natural language, leading to more contextually appropriate captions.

6.3. CLIP (`openai/clip-vit-base-patch32`) for Text Retrieval (Text-to-Caption)

6.3.1. Justification for Text Retrieval

- **Semantic Text Embeddings**: CLIP's text encoder is a powerful and highly effective text embedding model. It transforms natural language into dense vectors where semantic similarity is preserved as vector proximity. This is ideal for comparing a textual query against a collection of textual captions.
- **Consistency Across Modalities**: The most significant advantage of reusing CLIP for text retrieval is its shared embedding space. The text embeddings generated by CLIP (for both query and captions) exist in the exact same vector space as the image embeddings generated by CLIP. This means that if a user searches for "blue graphic T-shirt," and a generated caption "A blue T-shirt with a graphic print" is a good match, both will be semantically close in this shared space. This consistency is crucial for a unified multimodal system and could even enable hybrid retrieval approaches where both visual and textual features contribute to ranking.
- **No New Model Overhead**: Reusing an already loaded and utilized model (`CLIPModel`) minimizes memory footprint and simplifies the overall system architecture compared to introducing a separate, dedicated text embedding model (e.g., from the MTEB Leaderboard, like a Sentence Transformer model), though such models could offer higher precision for pure text-to-text similarity. For this project's scope, CLIP's versatility is a clear advantage.

6.4. FAISS (`faiss-cpu`) for Efficient Similarity Search

6.4.1. Model Description

FAISS (Facebook AI Similarity Search) is an open-source library developed by Meta (formerly Facebook) for efficient similarity search and clustering of dense vectors. It provides highly optimized implementations of various algorithms for finding the nearest neighbors of query vectors in a large database of vectors. It supports both exact nearest neighbor search (e.g., `IndexFlatL2`) and approximate nearest neighbor (ANN) search (e.g., `IndexIVFFlat`, `IndexHNSW`) for scalability to billions of vectors. The `faiss-cpu` variant specifically targets CPU execution, while `faiss-gpu` leverages NVIDIA GPUs for even faster computations.

6.4.2. Justification

- **Scalability for Embeddings**: With **1,993 T-shirt images**, each represented by a **512-dimensional vector**, a brute-force comparison for every query (calculating similarity with all 1,993 images) is feasible but not optimal for real-time performance, especially if the dataset were to scale to hundreds of thousands or millions of images. FAISS is purpose-built to handle such high-dimensional vector similarity searches efficiently.
- **Performance**: `IndexFlatL2` offers a high-performance, exact search. It is optimized through C++ implementations and SIMD instructions, making it much faster than a pure Python/NumPy loop for similarity calculations.
- **Integration with NumPy**: FAISS seamlessly integrates with NumPy arrays, which is the format in which embeddings are typically handled after being extracted from PyTorch tensors.

- **Industry Standard**: FAISS is a widely adopted and robust library in the industry for large-scale similarity search, making it a reliable and scalable choice for building retrieval systems.

By selecting these specific pre-trained models and auxiliary libraries, the project achieves a powerful and efficient multimodal information retrieval system for T-shirt images, adhering to the project's constraints and demonstrating best practices in leveraging modern AI tools.

7. Results and Discussion

The project aimed to develop a functional image retrieval and captioning system for T-shirt images, leveraging state-of-the-art pre-trained models. This section presents the key results from each task and discusses their implications.

7.1. Image Retrieval (Text-to-Image) Performance

The image retrieval system, powered by CLIP embeddings and a FAISS index, demonstrated effective text-to-image matching capabilities.

7.1.1. Qualitative Assessment

When provided with textual queries such as "green T-shirt" or "blue graphic T-shirt," the system consistently retrieved images that visually corresponded to the query. For instance, a query for "Green T-shirt" resulted in a display of **10 images**, with the top **5-7 results** typically showing T-shirts predominantly in green hues, some with minor graphics.

7.1.2. Strengths

- **Semantic Matching**: CLIP's ability to embed text and images into a shared semantic space is highly effective. Even for nuanced descriptions, it attempts to find visually relevant items.
- **Efficiency**: The pre-computation of **1,993 image embeddings** (each **512-dimensional**) and their indexing in FAISS (an `IndexFlatL2` in this case) allowed for near real-time retrieval once the system was set up. The embedding generation took approximately **5-10 minutes** on a typical CPU, but individual queries were answered within milliseconds.

7.1.3. Limitations

- **Specificity Trade-offs**: For very generic queries (e.g., "T-shirt"), the results might be broad. For highly specific but visually subtle features (e.g., "organic cotton T-shirt"), CLIP might struggle without explicit visual cues related to "organic cotton."
- **Dependence on Training Data**: While zero-shot, CLIP's performance is inherently tied to the diversity and quality of its pre-training data. If certain T-shirt features or styles were underrepresented in its training, retrieval for those might be less precise.
- **Ambiguity in Queries**: Natural language queries can be ambiguous. "Red T-shirt" might retrieve both bright red and maroon T-shirts, depending on the visual distribution in the dataset and CLIP's learned boundaries.

7.2. Image Captioning Performance

The BLIP model successfully generated descriptive captions for the T-shirt images, enriching the dataset with valuable textual metadata.

7.2.1. Caption Quality

The `Salesforce/blip-image-captioning-base` model produced generally accurate and coherent captions. For instance, for an image of a black T-shirt with white horizontal stripes, the model generated the caption "a p striped cotton t - shirt". Other examples included:

- Image of a plain blue V-neck T-shirt: "a blue t-shirt with a v-neck"

- Image of a grey T-shirt with a large white skull graphic: ”a gray t-shirt with a skull on the front”
- Image of a yellow T-shirt with text ”HELLO” on it: ”a yellow t-shirt with the word hello on it”

7.2.2. Strengths

- **Attribute Identification**: BLIP demonstrated an ability to identify key attributes such as color, pattern, neckline, presence of text/graphics, and even implied material (”cotton”).
- **Fluency**: The generated captions were grammatically sound and read naturally, making them suitable for human interpretation and subsequent text-based retrieval.
- **Efficiency**: Generating captions for **1,993 images** took approximately **30-40 minutes** on a CPU, which is acceptable for a one-time pre-computation.

7.2.3. Limitations

- **Missing Fine-grained Details**: While good, captions sometimes missed very specific or subtle details, such as brand logos, intricate patterns, or the specific context of a graphic if it wasn't a universally recognized object. For example, a band T-shirt might be captioned simply as ”a black t-shirt with a graphic.”
- **Commonality Bias**: Models like BLIP tend to generate captions reflecting the most common interpretations of visual features seen in their training data. This means rare or highly unique T-shirt designs might receive more generic captions.
- **Computational Cost**: Generating captions is more computationally intensive than just extracting embeddings for retrieval. Pre-computing is essential.

7.3. Text Retrieval (Text-to-Caption) Performance

The text retrieval system, using CLIP to compare textual queries against generated captions, demonstrated effective semantic search within the descriptive metadata.

7.3.1. Qualitative Assessment

Given a query like ”black T-shirt” or ”shirt with a skull design,” the system successfully retrieved images whose generated captions were most semantically similar to the query. For example, a search for ”black T-shirt” would likely return images whose captions included ”black t-shirt,” ”a t-shirt that is black,” or ”black top.”

7.3.2. Strengths

- **Semantic Search**: By embedding both query and captions into CLIP's semantic space, the system could find relevant items even if the exact keywords weren't present in the caption. For instance, a query ”casual top” might still retrieve captions mentioning ”t-shirt.”
- **Leveraging Rich Metadata**: This task highlighted the value of the image captioning module. The automatically generated captions became a searchable textual database, enabling queries that directly target the semantic content of the images.
- **Consistency with Image Retrieval**: Reusing CLIP ensured that the text embedding space for captions was consistent with the image embedding space for direct image retrieval, allowing for potential future hybrid approaches.

7.3.3. Limitations

- **Dependency on Caption Quality**: The performance of text retrieval is directly proportional to the quality and specificity of the generated captions. If BLIP produces a generic caption like ”a piece of clothing” for a unique T-shirt, the text retrieval system will be limited by that generic description.

- **Single-modality Comparison**: While powerful, this particular implementation of text retrieval solely relies on the captions. It doesn't incorporate the raw visual features for re-ranking. Images might have subtle visual cues missed by the captioning model that could still be relevant.
- **Lack of Reranking**: For improved precision, especially for the top results, a dedicated reranking model (e.g., cross-encoders from MTEB Leaderboard) could be integrated. These models would take the query and a retrieved caption pair, and score their relevance more deeply.

7.4. Overall System Performance and Integration

The integrated system effectively showcased the capabilities of multimodal information retrieval. The offline pre-computation of embeddings and captions for all **1,993 T-shirt images** forms the backbone of the efficient online retrieval.

7.4.1. Efficiency

The overall approach is highly efficient for inference. Once the initial embeddings and captions are generated (which are one-time, offline processes), subsequent queries yield results very quickly.

7.4.2. Modularity

The clear separation of tasks (image retrieval, captioning, text retrieval) into distinct modules, each utilizing specialized pre-trained models, demonstrates a robust and scalable architecture.

7.4.3. Value Proposition

For a business dealing with a large catalog of visual products like T-shirts, this system offers immense value:

- **Enhanced Searchability**: Customers can find products using natural language, improving user experience.
- **Automated Metadata Generation**: Reduces manual effort in cataloging and describing products.
- **Improved Product Discovery**: Facilitates finding relevant items even with abstract or non-keyword-based queries.

The project successfully achieved its goal of providing hands-on experience with information retrieval and the application of state-of-the-art pre-trained models from Hugging Face, confirming their effectiveness in real-world multimodal scenarios.

8. Conclusion and Future Directions

This project has successfully designed and implemented a comprehensive T-shirt image retrieval and captioning system, entirely based on the powerful capabilities of pre-trained models available through Hugging Face. By strategically employing CLIP for embedding images and text into a shared semantic space, and BLIP for generating descriptive captions, the system effectively addresses the core challenges of multimodal information retrieval.

The implementation demonstrated that:

- **CLIP**'s contrastive learning is highly effective for direct text-to-image retrieval, enabling semantic matching between textual queries and visual content. The integration with **FAISS** ensures that this retrieval is computationally efficient and scalable for growing datasets.
- **BLIP**'s generative capabilities provide a robust solution for automated image captioning, creating valuable textual metadata from purely visual input. These captions are natural-sounding and capture key attributes of the T-shirt images.
- Leveraging generated captions with **CLIP**'s text encoder enhances text retrieval by enabling semantic search based on the descriptive metadata, offering a powerful complement to direct visual search.

The project adhered to the critical constraint of not training models from scratch, thereby highlighting the immense utility and accessibility of pre-trained models in accelerating AI application development. The modular architecture and clear documentation ensure reproducibility and ease of understanding for anyone wishing to build upon this foundation.

8.1. Limitations and Areas for Improvement

While the system is robust and effective, several areas could be explored for further enhancement:

- **Fine-Grained Retrieval and Captioning**: Although the base models perform well, they might miss very subtle details or highly domain-specific nuances (e.g., differentiating between specific types of fabric textures, brand logos, or very intricate graphic styles). Fine-tuning a pre-trained model on a smaller, domain-specific dataset (if allowed by project constraints and resources) could improve performance for such details.
- **Hybrid Retrieval and Re-ranking**: The current system allows for either direct text-to-image retrieval (via CLIP's image embeddings) or text-to-caption retrieval (via CLIP's text embeddings of captions). A more advanced system could implement a **hybrid retrieval approach**, combining scores from both modalities to provide a more holistic relevance ranking. Furthermore, integrating a **reranking model** (e.g., a cross-encoder model from the MTEB Leaderboard) after an initial retrieval step could significantly boost the precision of the top-K results by performing a deeper, joint analysis of the query and the retrieved items/captions.
- **Scalability to Billion-Scale Data**: For extremely large datasets (millions or billions of images), transitioning from `faiss.IndexFlatL2` to Approximate Nearest Neighbor (ANN) FAISS indices (e.g., `IndexIVFFlat`, `IndexHNSW`) or integrating with specialized vector databases like ChromaDB or Elasticsearch's vector search capabilities would be essential for maintaining real-time query performance.
- **User Interface and Feedback Loop**: A production-ready system would benefit from a user-friendly interface. Implementing a feedback mechanism where users can mark results as relevant or irrelevant could also be used to gather data for future model improvements.
- **Quantifying Performance**: For a rigorous evaluation, quantitative metrics such as **Recall@K**, **Precision@K**, **Mean Average Precision (mAP)**, and **F1-score** would be computed, requiring a pre-defined ground truth set of relevant images for specific queries, or human annotations.

8.2. Broader Implications

This project underscores the transformative impact of open-source pre-trained models on the field of information retrieval:

- **Democratization of AI**: Complex multimodal AI capabilities, once exclusive to large research labs, are now accessible to a broader audience, enabling rapid prototyping and deployment.
- **Efficiency in Development**: The ability to leverage pre-existing knowledge embedded within these models drastically reduces the time and resources required to build powerful AI applications.
- **New Paradigms in Search**: Moving beyond keyword-based search to semantic, multimodal search opens up new avenues for how users interact with and discover information, particularly in visually rich domains like e-commerce, digital libraries, and social media.

In conclusion, the T-shirt image retrieval and captioning system serves as a practical demonstration of modern information retrieval principles. It exemplifies how carefully chosen pre-trained models can form the backbone of effective and scalable AI solutions, paving the way for more intuitive and intelligent interactions with digital content.