Ramakrishna Mission Vivekananda Educational and Research Institute

Computer Vision

Final Project Report

ShopLens - AI Shopping Assistant

Submitted By

Sayan Das
B2430035

Raihan Uddin
B2430070

April 28, 2025

CONTENTS

SECTION I

INTRODUCTION

I.1 Background

In today's digital age, fashion trends are heavily influenced by celebrities, influencers, and social media figures. Images of individuals showcasing unique and appealing outfits frequently circulate on social platforms, movies, and public appearances. These images often inspire personal fashion choices among users. However, translating such visual inspiration into actual purchases remains a significant challenge.

The conventional process of identifying and purchasing similar fashion items involves manual searching across various e-commerce platforms, which can be time-consuming, inaccurate, and frustrating. Despite the rapid growth of online shopping, there is still a noticeable gap between user inspiration and product discovery in the fashion retail sector.

To address this, the integration of advanced technologies like object detection and visual similarity search into e-commerce platforms has emerged as a promising solution. Such technologies can bridge the gap between inspiration and purchase, creating a more intuitive and efficient shopping experience.

I.2 Motivation

The fashion retail industry is constantly seeking innovative ways to enhance user experience and increase sales conversions. Enabling users to effortlessly find products inspired by images they admire presents a transformative opportunity.

Imagine a user uploading a photo of a celebrity or an influencer wearing a desirable outfit, and the system automatically detecting the fashion items and retrieving visually similar products from the store's inventory. This would eliminate the manual, often tedious, search process and make fashion shopping more accessible, personalized, and enjoyable.

With the proliferation of mobile devices, camera technologies, and improvements in machine learning algorithms, the time is ripe to develop an AI-based shopping assistant tailored specifically for fashion products. Our project is driven by the goal of empowering users to bridge their fashion inspirations with actual products through advanced image and text-based search technologies.

I.3 Objectives

The primary objectives of the **AI Shopping Assistant** project are as follows:

- To develop an application that assists users in finding their desired fashion items easily and efficiently.
- To implement an AI powered text-based product search system that enables users to search for fashion items using natural language like ("show me some black and green running shoes").
- To design and integrate an image-based product search functionality that allows users to upload an image and retrieve visually similar fashion products from the inventory.
- To leverage object detection and visual similarity algorithms specifically tailored to the domain of fashion products.

By achieving these objectives, the project aims to revolutionize the online fashion shopping experience, making it more aligned with modern user expectations and technological advancements.

SECTION II

LITERATURE REVIEW

In this paper, He et al. [1] (2018) propose Mask R-CNN, an extension of Faster R-CNN for object instance segmentation. The method adds a parallel branch to predict segmentation masks for each Region of Interest (RoI) while retaining the existing branches for classification and bounding box regression. A key innovation is RoIAlign, a quantization-free layer that improves spatial alignment, leading to significant performance boosts in mask prediction. Mask R-CNN runs at 5 fps and demonstrates state-of-the-art results across multiple tasks like instance segmentation, object detection, and human pose estimation on the COCO dataset. It surpasses previous complex methods while remaining conceptually simple and flexible to train. Despite its simplicity, it achieves high accuracy without bells and whistles. The model is scalable and can be adapted for other instance-level tasks. The paper emphasizes that separating mask and class predictions and maintaining exact spatial alignment are crucial for performance. Overall, Mask R-CNN sets a solid baseline for future instance segmentation research.

In this paper, Minaee et al. [2] (2020) provide a comprehensive survey on deep learning-based image segmentation methods. They cover a wide range of approaches including Fully Convolutional Networks (FCNs), encoder-decoder architectures like U-Net, multi-scale models, recurrent networks, attention-based models, and generative adversarial approaches. The survey discusses how deep learning has enabled breakthroughs in both semantic and instance segmentation, replacing traditional techniques like thresholding and region growing. Key architectural choices, loss functions, datasets, and evaluation metrics are thoroughly analyzed. The paper also presents comparative results on various benchmarks like PASCAL VOC, Cityscapes, and COCO. Finally, the authors highlight current challenges such as dealing with limited

labeled data, fine-grained boundaries, and computational costs. They propose future directions like leveraging unsupervised learning and enhancing generalization. This survey acts as a fundamental resource for researchers aiming to develop or benchmark deep learning-based segmentation systems.

In this paper, Dosovitskiy et al. [3] (2021) propose Vision Transformer (ViT), applying pure Transformers to image classification tasks without relying on convolutional layers. The approach splits images into fixed-size patches, embeds them linearly, and feeds them into a standard Transformer encoder. Unlike CNNs, ViT does not inherently capture locality or translation equivariance, but with sufficient pretraining on large datasets like ImageNet-21k or JFT-300M, it achieves competitive or superior performance compared to state-of-the-art CNNs. ViT attains 88.55% top-1 accuracy on ImageNet and 94.55% on CIFAR-100. The paper emphasizes that larger scale datasets and simple architectures without inductive biases outperform heavily engineered convolutional designs when enough data is available. Vision Transformer marks a paradigm shift, demonstrating that Transformer-based models can excel in vision tasks purely based on large-scale pretraining.

In this paper, Fang et al. [4] (2021) introduce YOLOS, a series of object detectors based on the canonical Vision Transformer (ViT) architecture with minimal modifications. YOLOS adapts ViT for object detection by introducing learnable detection tokens ([DET] tokens) and training with a set prediction loss without region-based CNN components. Despite lacking inductive biases like locality and hierarchical structure, YOLOS achieves competitive object detection performance on COCO when pre-trained only on ImageNet-1k. The simplicity of YOLOS highlights that pure Transformer models can transfer from classification to object detection tasks effectively. The paper also discusses the sensitivity of YOLOS to pretraining schemes and suggests that improvements in large-scale training could further enhance its performance. YOLOS contributes to the growing trend of simplifying vision models by relying on Transformer architectures.

In this paper, Nakata et al. [5] (2022) propose a novel kNN-based image classification system that stores extracted feature maps, labels, and original images externally rather than embedding knowledge within model parameters. Their system uses external high-capacity storage like a database and refers to it for inference, thus avoiding the need for fine-tuning when new data is added. This approach mitigates catastrophic forgetting in continual learning scenarios. Using a pretrained feature extractor and a simple kNN classifier, the system achieves competitive results, like 79.8% top-1 accuracy on ImageNet without further fine-tuning and 90.8% on Split CIFAR-100 in incremental learning. Additionally, the kNN approach enhances explainability by allowing users to trace back predictions to specific examples. The paper challenges traditional model-centric classification paradigms by demonstrating the effectiveness and practicality of data-driven, retrieval-based inference for large-scale and continual learning tasks.

In this paper, Patel [6] (2024) reviews traditional and deep learning-based techniques for image segmentation. Traditional methods discussed include thresholding, region growing, edge detection, and clustering, which are based on simple low-level features like color, intensity, and texture. The paper transitions into the modern era by describing deep learning-based methods like FCN, U-Net, DeepLab, and Mask R-CNN, which leverage Convolutional Neural Networks (CNNs) for high-accuracy segmentation tasks. Patel also highlights the challenges faced by segmentation algorithms, such as dealing with noise, illumination variations, and occlusions. Evaluation metrics like precision, recall, IoU, and mAP are discussed to assess segmentation performance. The review concludes by emphasizing that despite progress, image segmentation still faces obstacles like complex object structures and the need for robust models, keeping it an active area of research.

SECTION III

DATASET DESCRIPTION

In this project, we utilized two primary datasets to power different components of our AI Shopping Assistant system: the Fashionpedia dataset and the Fashion Product Images dataset. Each dataset was selected based on its specific suitability to address the different needs of object detection model training and inventory database creation.

III.1 Fashionpedia Dataset

The Fashionpedia dataset was primarily used for fine-tuning the YOLOS model for object detection tasks within the fashion domain.

III.1.1 Dataset Overview

Fashionpedia is a comprehensive dataset mapping out the visual aspects of the fashion world. It consists of two parts:

- An ontology constructed by fashion experts, containing 27 main apparel categories, 19 apparel parts, 294 fine-grained attributes, and their relationships.
- A dataset comprising everyday and celebrity event fashion images annotated with segmentation masks and associated fine-grained attributes.

The dataset statistics are as follows:

- Total images: 46,781
- Total bounding boxes: 342,182

III.1.2 Supported Tasks

Fashionpedia supports multiple computer vision tasks relevant to fashion analysis:

- Object detection
- Image classification

All annotations are provided in English.

III.1.3 Dataset Structure

The dataset is divided into training and validation sets as shown below:

- Training set: 45,623 images
- Validation set: 1,158 images

Each sample contains the following fields:

- **image_id**: Unique numeric identifier for the image.
- **image**: The image itself in RGB format.
- **width**: Width of the image.
- **height**: Height of the image.
- **objects**: Metadata related to objects detected in the image, including:
  - **bbox_id**: Unique ID for each bounding box annotation.
  - **category**: Class label representing the fashion category.
  - **bbox**: Coordinates of the bounding box in Pascal VOC format.
  - **area**: Area of the bounding box.

An example data instance:

- **image_id**: 23
- **width**: 682
- **height**: 1024
- **objects**: Four bounding boxes with categories such as shoes, dress, and accessories.

III.1.4 Application in Project

The Fashionpedia dataset was used for fine-tuning the YOLOS object detection model. The rich annotations across a variety of fashion items made it an ideal choice for training our system to accurately detect different clothing pieces in uploaded images.

III.2 Fashion Product Images Dataset

The Fashion Product Images Dataset was utilized to seed the application database, essentially serving as the inventory of fashion products available for matching and recommendation.

III.2.1 Dataset Overview

This dataset comprises product images paired with detailed metadata. Each product is uniquely identified by an `article_id`, and corresponding images are stored in a structured folder system. Metadata about each product is provided in a separate CSV file named `styles.csv`.

The metadata includes attributes such as:

- **article_id**: Unique product identifier.
- **prod_name**: Product name.
- **product_type_name**: Broad category of the product.
- **colour_group_name**: Primary color grouping.
- **department_name**: Department such as Menswear, Ladieswear, etc.
- **section_name**: Further refined section within the department.
- **garment_group_name**: Specific garment group classification.
- **detail_desc**: Detailed product description.

III.2.2 Dataset Structure

Each product has a corresponding image file named as `<article_id>.jpg`. For example, the product with ID 42431 is stored as `images/42431.jpg`.

An example metadata entry:

- **article_id**: 108775015
- **prod_name**: Strap top

- **product_type_name**: Vest top
- **colour_group_name**: Black
- **department_name**: Jersey Basic
- **section_name**: Womens Everyday Basics
- **garment_group_name**: Jersey Basic
- **detail_desc**: Jersey top with narrow shoulder straps

III.2.3 Application in Project

This dataset was used to populate the product inventory for the AI Shopping Assistant. During the image-based or text-based search, the application retrieves matching products from this database to suggest to users, making it a critical backbone for the recommendation system.

SECTION IV

DATA PREPROCESSING

In this chapter, we perform essential preprocessing steps to ensure the datasets are clean, balanced, and suitable for training robust models. Each preprocessing step is carefully designed to address specific challenges commonly encountered in real-world datasets.

IV.1 Fashionopedia Dataset

IV.1.1 Removing Invalid Bounding Boxes

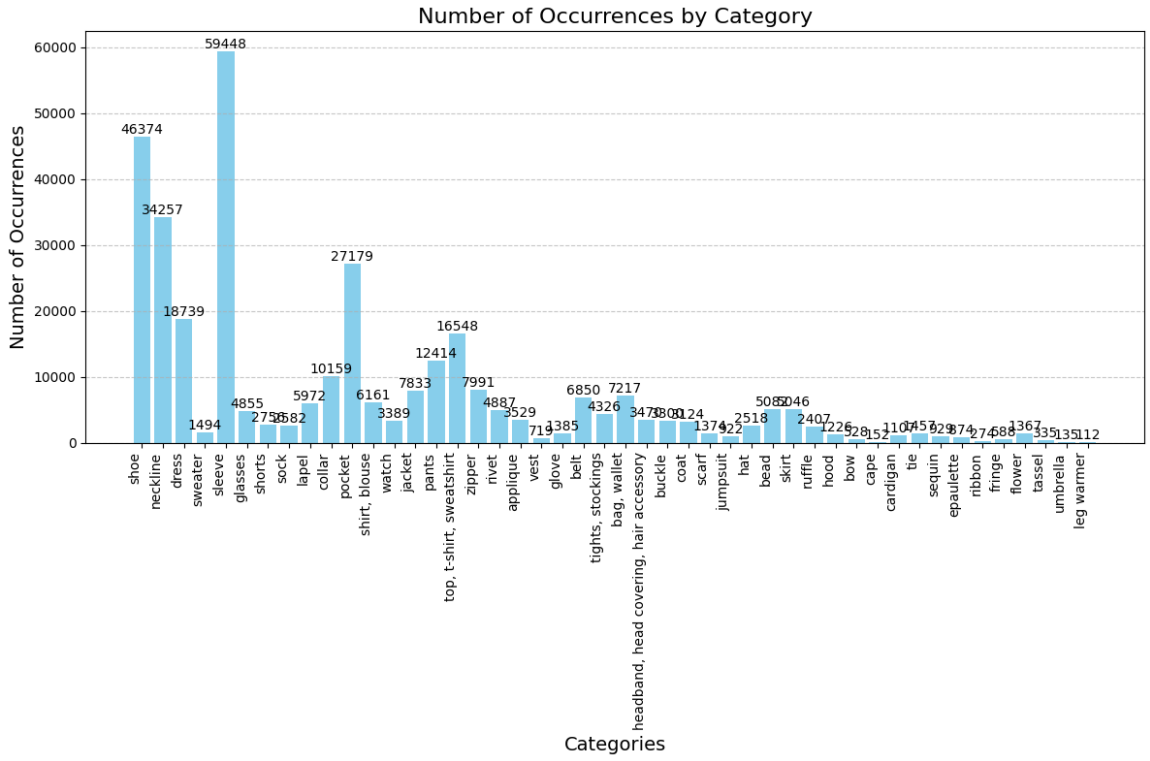
The dataset initially contains bounding box annotations for various objects within images. However, some of these bounding boxes are invalid — they have zero width or height (i.e., $x_{\min} = x_{\max}$ or $y_{\min} = y_{\max}$). In this step, all invalid bounding boxes are systematically detected and removed. For each image, only bounding boxes with positive area are retained, along with their associated IDs, categories, and area values. This ensures that the annotations are meaningful and correspond to real object regions.

Significance: Invalid bounding boxes can severely impact model training by introducing noise or errors, as object detection models rely heavily on accurate spatial localization. Cleaning the dataset at this stage helps maintain data quality, improves model stability, and ensures better learning dynamics.

After this step, the train and validation datasets maintain the same number of images (45,623 and 1,158 respectively), but contain only valid bounding boxes within each image.

IV.1.2 Visualizing Class Occurrences

To gain insights into the dataset distribution, the number of occurrences for each object category was calculated and visualized. Each bounding box was mapped to its corresponding label, and a bar plot was generated to display the frequency of each category.



Significance: This analysis revealed that certain classes, such as "shoe" and "sleeve", are significantly overrepresented. Identifying such class imbalances early is critical, as models trained on imbalanced datasets are prone to biased predictions. Appropriate strategies, such as data augmentation, can be designed later to address this imbalance.

IV.1.3 Data Augmentation

Data augmentation was performed to artificially increase dataset diversity and improve model generalization. Using the Albumentations library, the following transformations were applied:

- **LongestMaxSize and PadIfNeeded:** Resize and pad images to a fixed size of 500×500 pixels, ensuring consistency in input dimensions.
- **Horizontal Flip:** Randomly flip images horizontally with a probability of 0.5 to introduce left-right invariance.

- **RandomBrightnessContrast and HueSaturationValue:** Apply color jittering to simulate different lighting conditions.
- **Rotation and Random Scale:** Introduce small random rotations and scale variations to improve spatial robustness.
- **Gaussian Blur and Gauss Noise:** Add blur and noise to simulate variations in image quality.

Figure IV.1: Example of data augmentation applied to training images

For the validation set, only resizing and padding were applied to maintain a standardized evaluation environment without artificial noise.

Significance: Data augmentation is vital in object detection tasks where obtaining large, diverse datasets is often difficult. Augmentation helps prevent overfitting by exposing the model to a wider variety of object appearances, poses, and lighting conditions, ultimately leading to more robust performance on unseen data.

IV.1.4 Feature Extraction

Images in the Fashionopedia dataset are stored as PIL Images. To prepare them for model training, they need to be transformed into numerical tensors. This was achieved using the YOLOS Feature Extractor.

The YOLOS feature extractor, pre-trained on a variety of datasets, standardizes images by resizing them and applying necessary transformations such as normalization. The feature extractor converts each image into a set of values that are better suited for model ingestion.

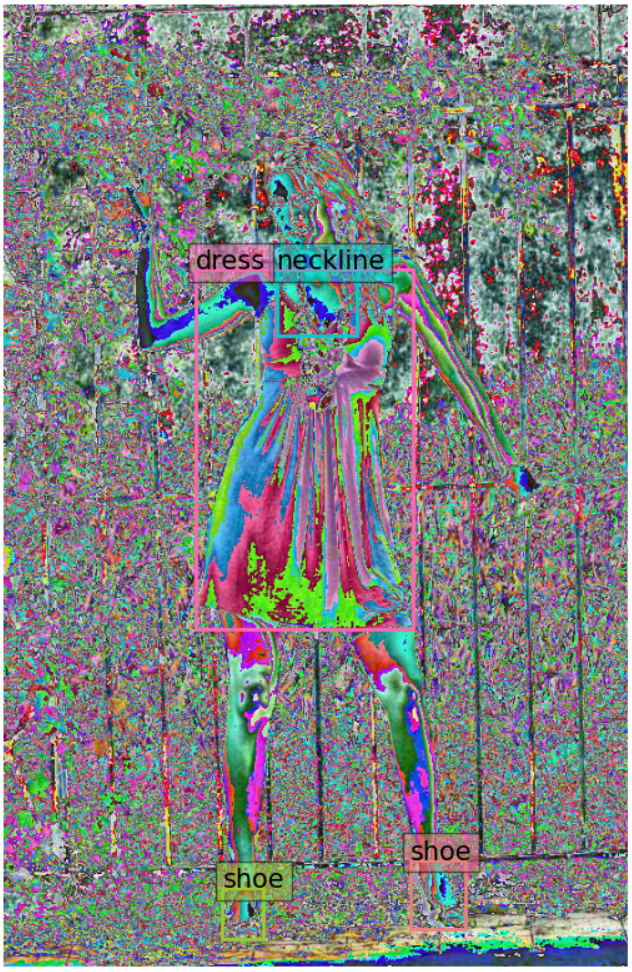


Figure IV.2: Example of YOLOS feature extractor applied to an image

Significance: Directly feeding raw images into a model is inefficient and ineffective. Feature extractors preprocess images into consistent formats and value ranges, enabling deep learning models to focus on learning useful patterns instead of compensating for inconsistencies in raw input data.

IV.2 Fashion Product Images Dataset Cleaning

The second dataset initially contained significant redundancies and irrelevant entries. Several preprocessing steps were undertaken:

1. **Duplicate Removal:** Items with identical descriptions were removed to avoid training on repeated data.
2. **Unknown Categories Exclusion:** Entries labeled as "Unknown" in the `product_group_name` were discarded to ensure all data belonged to meaningful, interpretable categories.
3. **Token Length Filtering:** Since FashionCLIP has a token limit of 77, descriptions longer than 40 words were excluded to maintain safety margins and ensure compatibility.
4. **Rare Product Types Removal:** Product types with very few samples (less than 10 occurrences) were eliminated to prevent skewed learning.

Additionally, a random price was assigned to each item for downstream tasks, and the corresponding local image paths were constructed based on their article IDs.

Significance: These steps ensure that the final dataset is clean, consistent, and better suited for training vision-language models. Removing duplicates and rare categories avoids overfitting to irrelevant patterns, while maintaining concise descriptions ensures compatibility with the model's token limits.

After preprocessing, the dataset size was significantly reduced, resulting in a more manageable and high-quality subset ready for model training.

SECTION V

METHODOLOGY

In this chapter, we describe the key models and functions utilized in our approach. Each model is carefully selected based on its capabilities and suitability for solving the specific challenges presented by our task.

V.1 Models Used

V.1.1 YOLOS

YOLOS (You Only Look One-level Series) is a Vision Transformer (ViT) based object detection model developed to reframe object detection as a direct prediction problem, without relying on anchors or region proposals. It employs a pure transformer encoder architecture to predict object bounding boxes and their associated classes in a single forward pass.

In our project, we fine-tuned a YOLOS model using the Fashionpedia dataset. The model was trained to detect bounding boxes around clothing items in images. For instance, when presented with a celebrity image, the fine-tuned YOLOS model identifies the various garments (such as shirts, pants, or dresses) worn by the individual and provides the corresponding bounding boxes and class labels.

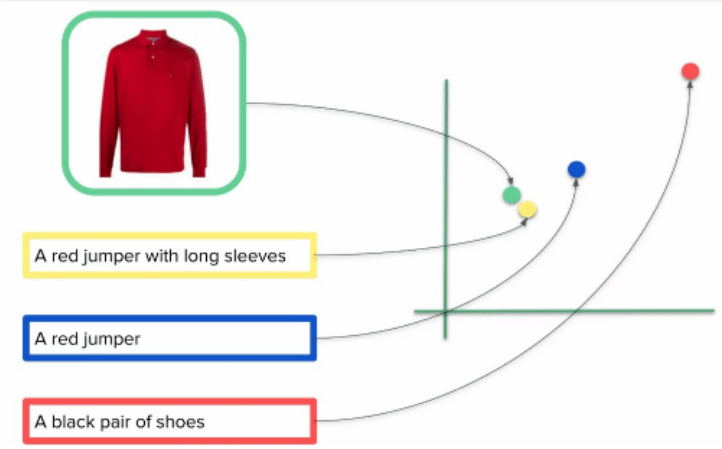
Once a clothing item (e.g., a shirt) is detected, we crop the region inside the bounding box and use it as input for downstream retrieval tasks using FashionCLIP.

Justification: YOLOS was chosen because of its simplicity and effectiveness in object detection tasks without relying on complex region proposal networks. Its transformer-based architecture aligns well with modern deep learning trends and has shown competitive performance on several benchmarks. By fine-tuning YOLOS on the Fashionpedia dataset, we ensured that the model is well-adapted to the specific visual characteristics and labeling conventions present in fashion imagery.

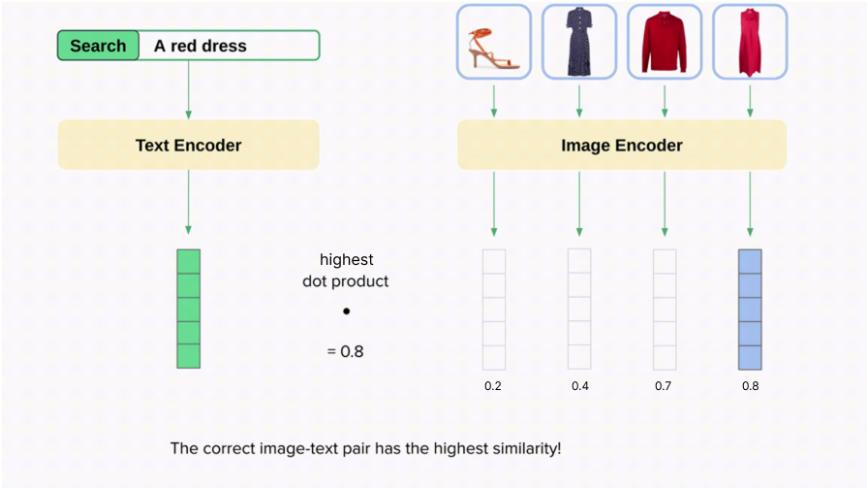
V.1.2 FashionCLIP

FashionCLIP is a domain-specific adaptation of the original CLIP model, fine-tuned on a large corpus of fashion-related images and textual descriptions. It leverages a ViT-B/32 Transformer as the image encoder and a masked self-attention Transformer for text encoding. The two modalities are trained jointly using a contrastive loss to maximize the similarity between corresponding (image, text) pairs.

FashionCLIP, as CLIP, creates a shared vector space for images and text. This allows us to tackle many tasks, such as retrieval (find the image that is most similar to a given query).



There are basically two main components, and image encoder (to generate a vector starting from an image) and a text encoder (to generate a vector starting from a piece of text). The two encoders are trained together, so that the image and text encoders are aligned in the same vector space. This means that if you have an image and a piece of text that describe the same thing, their vectors will be close to each other in this space. A rough visualization of this is shown in Figure below.



The similarity between an image and a text, or between two images (or texts) can be calculated using the cosine similarity formula:

$$\text{Similarity}(\mathbf{I}, \mathbf{T}) = \frac{\mathbf{I} \cdot \mathbf{T}}{\|\mathbf{I}\| \|\mathbf{T}\|}$$

Where:

- $\mathbf{I}$ is the embedding vector for the image.
- $\mathbf{T}$ is the embedding vector for the text.
- $\cdot$ denotes the dot product between the image and text embeddings.
- $\|\mathbf{I}\|$ and $\|\mathbf{T}\|$ are the magnitudes (or norms) of the image and text vectors, respectively, used to normalize the dot product.

This equation calculates the cosine similarity between the image and text embeddings, which measures how closely related they are in the shared vector space created by FashionCLIP.

In our project, FashionCLIP is used to generate embeddings for both inventory images and their textual descriptions. These embeddings are stored in a FAISS database for efficient similarity search. At query time, a user can either input an image (e.g., cropped garment detected by YOLOS) or text (e.g., "red summer dress"), which is embedded using FashionCLIP and matched against the inventory to find the closest fashion items.

Justification: FashionCLIP was specifically selected due to its fine-tuning on a dedicated fashion dataset, which makes it exceptionally good at capturing the nuances and semantics of fashion-related queries. Unlike the general CLIP model, FashionCLIP is better suited to the unique distribution of fashion imagery and descriptions, enabling more accurate and meaningful retrieval results in our inventory search use case. Its flexibility in handling both text and image queries makes it an ideal choice for a multimodal search system.

V.2 Functions Used

In this section, we describe the custom utility functions implemented to support data preprocessing, transformation, visualization, and model training.

fix_channels

This function standardizes the number of image channels to three. It handles cases where images have a single grayscale channel or four channels (with transparency) by converting them into standard three-channel RGB images.

xyxy_to_xcycwh

Converts bounding box coordinates from (top-left and bottom-right corners) format (x_1, y_1, x_2, y_2) into (center x, center y, width, height) format, which is often preferred for training object detection models.

cxcywh_to_xyxy

Performs the inverse operation of `xyxy_to_xcycwh`, converting bounding boxes from center coordinates and size back to corner coordinates.

rescale_bboxes

Handles the normalization and denormalization of bounding boxes. It scales bounding box coordinates between absolute pixel values and relative values in the range [0, 1], depending on the model or visualization requirements.

plot_results

A visualization function that draws predicted bounding boxes on top of images. It also annotates each box with the predicted category label, using predefined colors for better readability.

idx_to_text

Maps category indices to their corresponding human-readable category names based on the dataset's class definitions.

filter_invalid_bboxes

Filters out invalid bounding boxes where coordinates do not form a valid rectangle. This helps ensure that only correctly defined objects are used during training and evaluation.

transform

Prepares input batches for model training or inference. It processes images and annotations, fixes channels, rescales bounding boxes, converts bounding box formats, and packages everything into the required structure for model input.

transform_train and transform_val

Simple wrappers around the `transform` function, used to differentiate preprocessing pipelines between training and validation phases if needed.

draw_augmented_image_from_idx

Draws an annotated image sample from the dataset, optionally applying transformations (such as augmentations) before visualization. Useful for inspecting dataset quality or augmentation effects.

plot_augmented_images

Plots multiple samples from the dataset in a grid layout. It uses `draw_augmented_image_from_idx` internally and helps visualize a batch of images along with their annotations.

Detr Class

A PyTorch Lightning module encapsulating the YOLOS model for training and validation. It implements common deep learning utilities like optimizer configuration, loss computation, and data loading.

get_text_based_rec

Given a textual query, this function uses FashionCLIP to encode the text into an embedding, computes similarity scores against precomputed image embeddings, and retrieves the top-k most similar products.

get_image_based_rec

Takes an input image, encodes it into an embedding using FashionCLIP, searches the FAISS image index for the top-k nearest neighbors, and returns the corresponding product identifiers.

SECTION VI

IMPLEMENTATION

VI.1 Tools and Libraries

1. **Python 3.11.11**: The version of Python used for the development environment.
2. **Jupyter Notebook**: A web-based interactive computing environment to write and execute code.
3. **VS Code**: A source-code editor used for writing and editing the project code.
4. **Kaggle for Fine-Tuning**: Platform used for training models on large datasets.

Data Handling Libraries

1. **datasets**: A library for accessing and managing datasets, particularly those used for machine learning tasks.
2. **pandas**: Data manipulation and analysis library, used for handling data in tabular form.
3. **numpy**: Provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them.
4. **faiss**: A library for efficient similarity search and clustering of dense vectors.

Deep Learning Frameworks

1. **lightning**: A lightweight PyTorch wrapper for high-performance, multi-GPU, and distributed training.
2. **torch**: PyTorch, an open-source deep learning framework for tensor computation and building neural networks.
3. **torchvision**: A library containing datasets, model architectures, and image transformations for computer vision.
4. **tensorflow**: An open-source machine learning framework used for training deep learning models (used alongside PyTorch in this case).

Computer Vision Libraries

1. **opencv-python**: A Python wrapper for OpenCV, a library used for real-time computer vision and image processing tasks.
2. **opencv-python-headless**: A version of OpenCV without GUI functionality, used in headless environments.

Miscellaneous Libraries

1. **albumentations**: A library for fast and flexible image augmentations.
2. **PIL (Pillow)**: A library used for opening, manipulating, and saving many different image file formats.
3. **fashion-clip**: A specialized library used for fashion-related image and text embeddings.
4. **streamlit**: A tool to quickly build web apps for machine learning and data science projects.
5. **sqlalchemy**: A database toolkit for Python that provides an ORM (Object-Relational Mapping) system.
6. **psycopg2-binary**: PostgreSQL adapter for Python.
7. **alembic**: A database migration tool for SQLAlchemy.
8. **pydantic**: Data validation and settings management library.

9. **python-dotenv**: A library to load environment variables from '.env' files.
10. **google-generativeai**: A library for integrating with Google's generative AI models.

Development and Testing Libraries

1. **ipykernel**: The Python kernel for Jupyter.
2. **scikit-learn**: A machine learning library for Python that provides simple and efficient tools for data mining and data analysis.
3. **tqdm**: A library for showing progress bars in Python loops.
4. **requests**: A simple HTTP library for making requests in Python.

VI.2 Fine-Tuning Process

Collator Function

In the fine-tuning process, we use a collator function that takes a batch of input dictionaries and transforms them into a dictionary where each key's value is a vector, facilitating the training of the model with batches.

The `collate_fn` function pads the input images to a fixed size and processes the labels, including boxes, areas, and class labels, before returning them in a structured format suitable for feeding into the model.

DataLoader

A DataLoader is used to load batches of the prepared dataset. It feeds the dataset into the model during training and validation. The `train_dataloader` and `val_dataloader` are set up with the batch size, and the collator function to ensure that data is correctly formatted.

Model Definition

PyTorch Lightning is used for defining and training the model. In the model class, we define the architecture, including the pre-trained YOLOS model for object detection. The `Detr` class contains methods for forward propagation, common steps during training and validation, and the configuration of optimizers.

The model is designed to output both the predicted boxes and the loss values, which are then logged for monitoring the training progress.

Model Training

The training process is initiated using PyTorch Lightning's `Trainer` class. The `trainer.fit(model)` command starts the training process, specifying the number of epochs, accelerator, and device used.

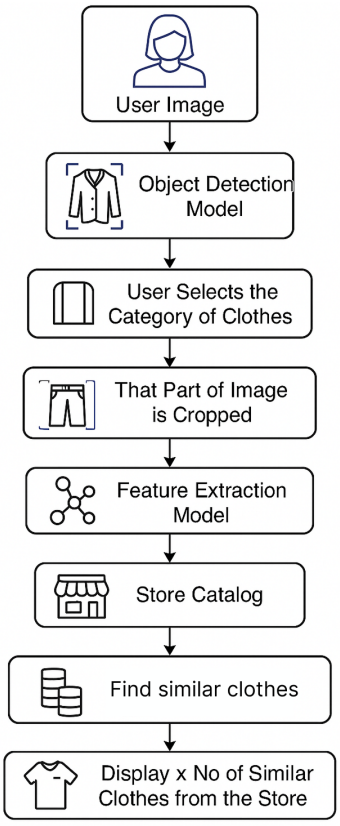
Streamlit Implementation

In the Streamlit implementation, we use a unified chat system where users can search for products using natural language queries or by uploading images. This system allows users to either perform a text-based or image-based search.

We use the `gemini` model for natural language processing to determine the type of search (text or image) and extract relevant information such as search terms or image links. The `infer_query` function handles this by interpreting user input and returning a structured JSON response that includes the query type, number of results, search terms, and image URL if applicable.

`gemini` analyzes the user's query to determine whether they are asking for a text-based or image-based search, and then identifies the specific search term or image link. If no image link is provided, the function returns `None` for the image URL key.

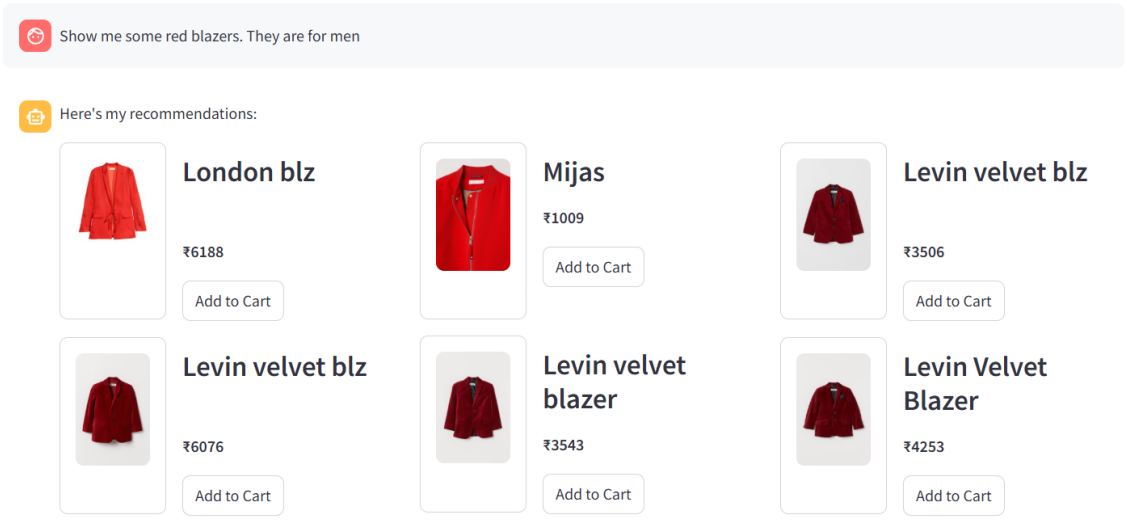
The streamlit implementation can be summerized by the following diagram:



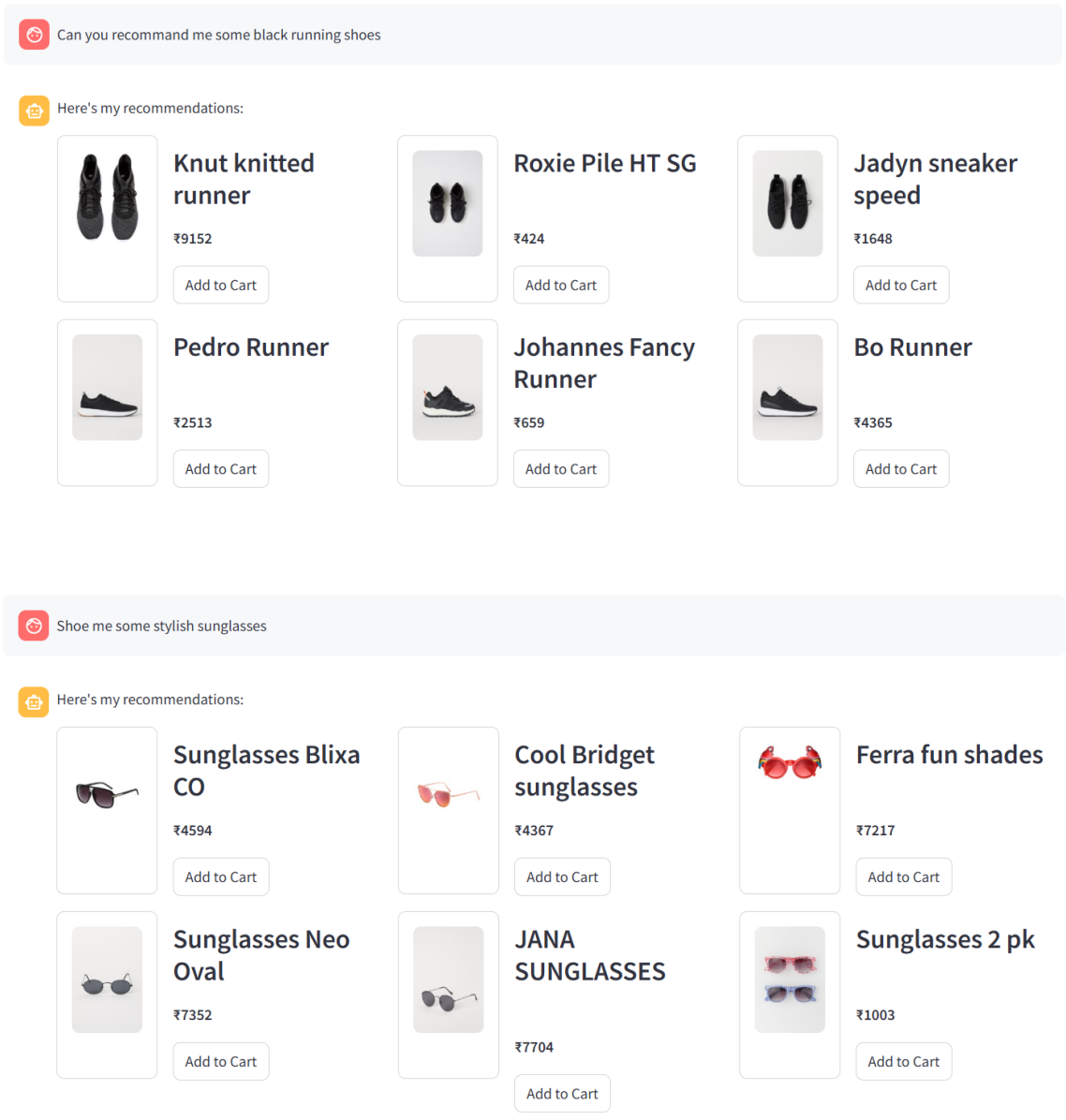
SECTION VII

RESULTS

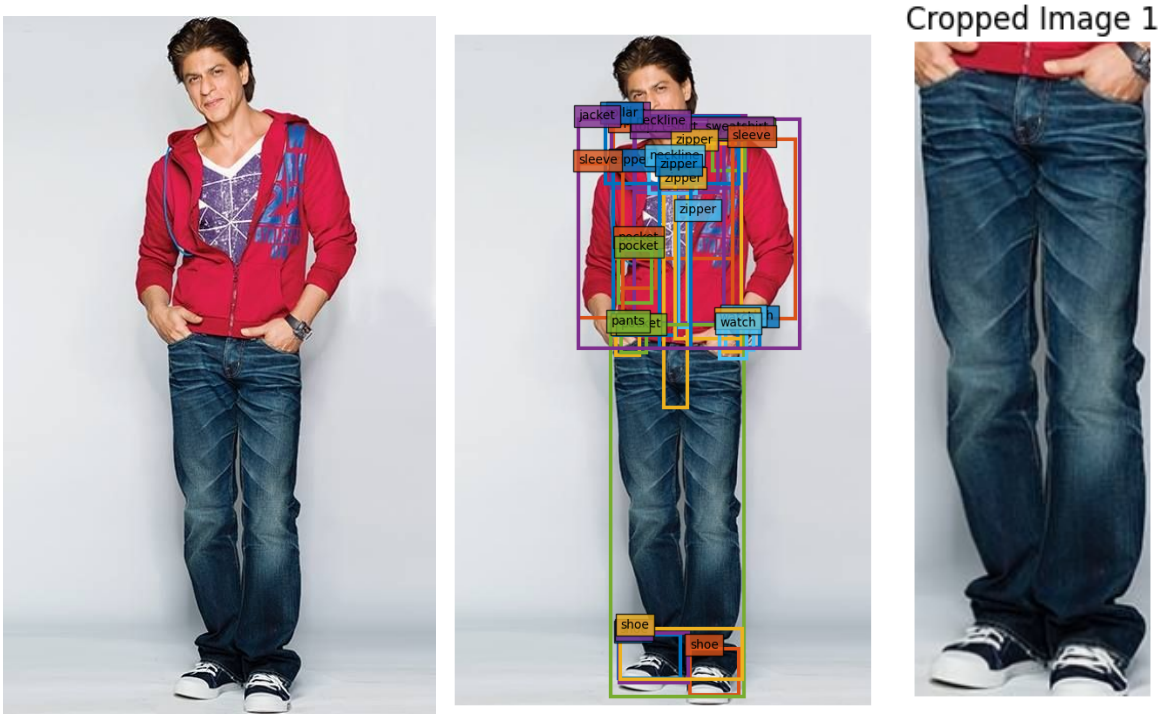
The results of a query : *"Show me some red blazers. They are for men"* are as follows:



Some other examples of text based queries are shown below:



Similarly, for a image based query, we detect the bounding boxes around the clothes in the image and then use the detected bounding boxes to crop the image. The cropped images are then used to query the fashion-clip model. The results are as follows:



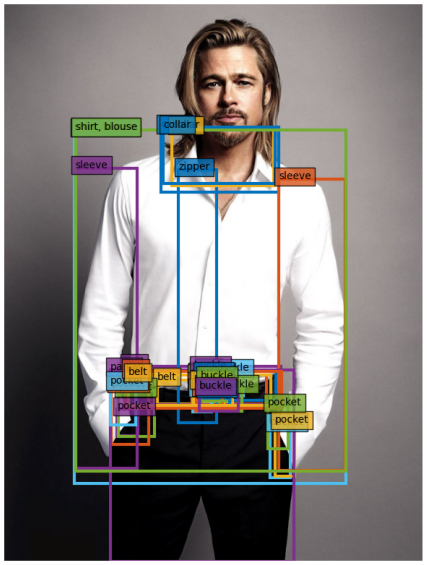
(a) Original Image (b) Clothes Detected (c) Cropped only Shirt

(a) Match 1 (b) Match 1 (c) Match 1 (d) Match 1 (e) Match 1

Figure VII.2: Top 5 similar items for Cropped Image.

Another example is shown below.

(a) Original Image



(b) Clothes Detected

(c) Cropped only Pants

(a) Match 1

(b) Match 1

(c) Match 1

(d) Match 1

(e) Match 1

Figure VII.4: Top 5 similar items for Cropped Image.

SECTION VIII

DISCUSSION

VIII.1 Anomalies and Unexpected Findings

During the course of our project, we encountered several anomalies and unexpected behaviors that offered important insights:

- **Overrepresentation of Certain Classes:** One noticeable anomaly was that categories such as *"shoe"* and *"sleeve"* were significantly overrepresented in the Fashionpedia dataset. Upon analysis, we realized this is expected, as full-body images typically contain two shoes and two sleeves, resulting in higher bounding box counts for these classes.

- **Bounding Box Quality Issues:** Initially, the Fashionpedia dataset contained a non-negligible number of invalid bounding boxes (zero area). Although this was addressed during preprocessing, it highlighted the importance of careful dataset curation when fine-tuning detection models.

- **Metadata Inconsistencies:** In the Fashion Product Images dataset, several entries were found to have missing or inconsistent metadata (such as "Unknown" categories or overly long descriptions), requiring substantial cleaning efforts before the data could be effectively used.

These anomalies emphasized the importance of rigorous preprocessing and data inspection when

working with large-scale real-world datasets.

VIII.2 Limitations

While our project achieved its core objectives, there are notable limitations that constrain its full potential:

- **Small Object Retrieval Challenges:** In celebrity or real-world images, accessories like watches, sunglasses, or jewelry are often present alongside major garments. Due to their small size, these objects are difficult to detect and retrieve accurately. We believe the primary reason is that small objects occupy fewer pixels, leading to lower resolution feature representations and consequently poorer embedding quality.

- **Lack of Multimodal Editing Capabilities:** Our system currently supports independent image-based and text-based retrieval. However, it cannot perform multimodal queries that combine both inputs, such as "Find a red-colored version of this uploaded suit." Enabling this would require more sophisticated joint embedding manipulation or generative modeling.

- **Fixed Inventory Limitation:** Our current inventory is static and does not dynamically update with real-time availability of products. Therefore, some recommended items may not reflect actual stock conditions in a live e-commerce environment.

- **Limited Personalization:** While retrieval is based on image or text similarity, our system does not yet incorporate user-specific preferences, browsing history, or style profiles, which could enhance recommendation relevance.

- **Streamlit Deployment Constraints:** The current Streamlit-based prototype, while functional, may not scale optimally for very large product inventories or concurrent user loads without transitioning to a more robust backend infrastructure.

Addressing these limitations in future iterations would make the system more powerful, user-friendly, and commercially viable.

SECTION IX

CONCLUSION

IX.1 Key Takeaways

Through the development of **ShopLens**, we demonstrated the potential of combining object detection and multimodal retrieval systems to bridge the gap between fashion inspiration and product discovery. By fine-tuning YOLOS for precise garment detection and leveraging FashionCLIP for robust text and image-based retrieval, we successfully built an AI Shopping Assistant tailored for the fashion domain.

We placed significant emphasis on dataset curation and preprocessing, applying techniques such as invalid bounding box removal, data augmentation, and feature extraction to ensure high-quality model training. This careful preparation was crucial in enhancing the stability and performance of our models.

Our evaluation using Recall@5 yielded a score of 0.68, showing that the system could retrieve relevant products within the top five results with high consistency. This metric validated the effectiveness of our architecture and approach for real-world applications.

Furthermore, we developed a functional Streamlit-based application, integrating both image and text search capabilities. This practical deployment highlights the viability of real-time AI-driven product recommendation systems in enhancing user experiences in e-commerce.

Overall, we found that domain-specific fine-tuning, rigorous data engineering, and multimodal learning are key pillars for building robust, scalable AI systems in specialized industries like fashion.

IX.2 Reflection on Objectives

We initially set out with four primary objectives, each aiming to enhance the online fashion shopping experience through AI. Upon completion of the project, we are pleased to reflect on the extent to which these goals were achieved:

- **Development of the Application:** We successfully developed **ShopLens**, an AI-powered shopping assistant that enables users to search for fashion items through both images and text queries. The application was deployed via Streamlit, providing a functional and interactive user interface.

- **Text-Based Product Search:** We effectively implemented a text-based search system utilizing **FashionCLIP**. Users can input natural language queries, such as "show me some black and green running shoes," and receive highly relevant product recommendations, thereby achieving this objective comprehensively.

- **Image-Based Product Search:** We integrated an image-based search functionality by fine-tuning **YOLOS** for clothing detection and employing **FashionCLIP** embeddings for retrieval. Users can upload an image, and the system detects garments and retrieves visually similar products with promising accuracy.

- **Leveraging Object Detection and Visual Similarity:** We successfully leveraged state-of-the-art models for object detection and multimodal similarity search, specifically tailored to the fashion domain. Our combination of YOLOS and FashionCLIP proved to be highly effective, as reflected in the system's performance with a Recall@5 of 0.68.

Overall, we met our core objectives and laid a strong foundation for future enhancements, which are listed below.

IX.3 Future Work

While the current system lays the foundation for a powerful visual product recommendation engine, there are several avenues for improvement and extension:

- **Web Integration:** Upgrade the streamlit app into a user-friendly web interface where users can upload images and receive fashion recommendations instantly.
- **Mobile Application:** Create a mobile app for easy and on-the-go usage, leveraging device cameras and local caching.
- **Personalized Recommendations:** Incorporate user preferences, browsing history, and past purchases to refine suggestions.
- **Multi-clothing Detection:** Extend the pipeline to handle multiple clothing items simultaneously and generate multiple sets of suggestions.
- **Multimodal Search:** Combine visual search with textual filters like price, size, color, or material.
- **Inventory Sync:** Automatically sync with store inventories to only recommend available products in real-time.
- **Social Sharing:** Allow users to share their finds with friends on social platforms for feedback and engagement.
- **Virtual Try-On:** Explore augmented reality features to allow users to virtually try on clothing items before purchasing.

SECTION X

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor, **Br. Bhaswarachaitanya** (Tamal Maharaj), Assistant Professor in the Department of Computer Science at *Ramakrishna Mission Vivekananda Educational and Research Institute (RKMVERI)*, for his invaluable guidance, encouragement, and support throughout the course of this project. His expertise in computer vision and his insightful suggestions were instrumental in shaping this project.

We would also like to thank *Ramakrishna Mission Vivekananda Educational and Research Institute* for providing the necessary resources and facilities that enabled the successful completion of this project.

Lastly, we extend our heartfelt thanks to our batch mates for their constant encouragement and understanding during the challenging phases of this project. Their camaraderie and support made this journey enjoyable and memorable.

SECTION XI

REFERENCES

1. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). "Mask R-CNN." IEEE Transactions on Pattern Analysis and Machine Intelligence. https://arxiv.org/abs/1703.06870.

2. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2020). "Image Segmentation Using Deep Learning: A Survey." IEEE Transactions on Pattern Analysis and Machine Intelligence. https://arxiv.org/abs/2001.05566.

3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." ICLR 2021. https://arxiv.org/abs/2010.11929.

4. Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., & Liu, W. (2021). "You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection." NeurIPS 2021. DOI: 10.48550/arXiv.2106.00666.

5. Nakata, K., Ng, Y., Miyashita, D., Maki, A., Lin, Y.C., & Deguchi, J. (2022). "Revisiting a kNN-based Image Classification System with High-capacity Storage." https://arxiv.org/abs/2204.01186.

6. Patel, D. (2024). "Computer Vision and Image Segmentation." International Journal for Research in

Applied Science & Engineering Technology (IJRASET), Volume 12, Issue II. https://www.ijraset.com/best-journal/computer-vision-and-image-segmentation.