## PROJECT REPORT ON

**Visual Product Recognition**

## SUBMITTED IN PARTIAL FULFILMENT

## OF

## THE REQUIREMENTS OF

## THE AWARDS OF THE

#### PG-Diploma in Big Data Analyst

**Offered By**

**C-DAC Hyderabad**

BY

**Pravin Ananda Kalse 230350325016**

**Rajan Bhumeshwar Katre 230350325019**

**Vijay Gopal Tiwari 230350325032**

**Ganesh Vasant More 230350325005**

**Sakshi Nitin Thakre 230350325022**

**Advanced Computing Training School  
C-DAC  
Hyderabad-501510**

**MARCH 2023**



Certificate

This is to certify that this is a bonafide record of project entitled **“Visual Product Recognition”**. (Pravin Kalse, Rajan Katre, Vijay Tiwari, Ganesh More, Sakshi Thakre) has completed project work as part of **Diploma in Advanced Secure Software Development (March 2023 Batch)**, a PG course offered by C-DAC Hyderabad. They have completed project work under the supervision of Mr Manish Bharti. Their Performance found to be good.

**Name of Project guide**

Mr Manish Bharti

**Date**  :

**Place**  : C-DAC, Hardware Park, HYDERABAD.

**ACKNOWKEDGEMENT**

Visual Product Recognition project has presented, an objective, a goal, a challenge of data security. This project marks the final hurdle that we tackle, of hopefully what would be one of the many challenges we have taken upon and am yet to take.

However, we could not have made it without the support and guidance from the following. Firstly I want to take this opportunity to have special thanks to our guide **Mr Manish Bharti** who helped us throughout this project by providing valuable guidance and advice as well as acquiring all components needed for this project to become a success.

(PG-DBDA MARCH 2023)

**Pravin Kalse 230350325016**

**Rajan Katre 230350325019**

**Vijay Tiwari 230350325032**

**Ganesh More 230350325005**

**Sakshi Thakre 230350325022**

|  |
| --- |
| **TABLE OF CONTENTS** |
| **TOPIC PAGE** |
| **Abstract I** |
| 1. **Introduction 6** |
| 1. **Literature Review 6** |
| **3. Problem Statement 7** |
| **4. Design 7** |
| **5. Description 8**  **5.1 Image Recognition vs Image Detection 9**  **5.2 How Does Image Recognition Works 9** |
| **6.Implementation 11**  **6.1 Step Involve**  **6.2 How to apply image recognition with python** |
| **7.Architecture 12** |
| **8.Training and implementation 14** |
| **9.Result and Evaluation 17** |
| **10 Future Scope 17** |
| **11.Conclusion 17** |

**Abstract:**

In this project, we address the challenge of recognizing products in user photos and matching them with corresponding products in seller photos. We propose an algorithm that utilizes object bounding boxes provided in user photos to identify desired products. Our approach aims to bridge the gap between user-generated images and seller representations to enhance product search accuracy.

Fashion is the way we present ourselves which mainly focuses on vision, has attracted great interest from computer vision researchers. It is generally used to search fashion products on online shopping sites to know the descriptive information of the product. The main aim of this project is to use deep learning (DL) and machine learning (ML) methods for the task of image-based product search. In this work, we used image scraping to collect the images from customer reviews. further we annotated these images, created labels and stored this data into a database (SQL).After the data is processed, we used YOLO: you only look once : unified, real-Time object detection to perform detection. The main goal of this process is to teach the system to draw boxes around products in pictures that users upload. Once the system identifies the products in these boxes, it will connect them to the right entries in a database. Each entry will have a link that takes users to where they can buy the product. The main purpose is to make the system smart enough to understand and match these pictures with the right products, so users can easily find and buy what they're looking for.

**1. Introduction**:

Automated product recognition has significant implications for e-commerce and inventory management. However, recognizing products in user photos, which often contain clutter and uncontrolled environments, presents a unique challenge. This project focuses on developing a solution to accurately match user-generated product images with seller representations. By leveraging object bounding boxes as search queries, our algorithm seeks to enhance the shopping experience by providing accurate product matches.

Enabling quick and precise search among millions of items on marketplaces is a key feature for e-commerce. The use of common text-based search engines often requires several iterations and can render unsuccessful unless exact product names are known. Image-based search provides a powerful alternative and can be particularly handy when a customer observes the desired product in real life, in movies or online media.

Our evaluation targets a real-case scenario where we use over 4k images for 10 products from real marketplaces. Example products include sandals and sunglasses, and their successful matching requires overcoming visual variations in images due changing viewpoints, background clutter, varying image quality and resolution

**2. Literature Review:**

Prior research has explored various methods for image recognition and matching. Deep learning techniques, especially Convolutional Neural Networks (CNNs), have shown promising results in image classification and object detection. Transfer learning, which involves fine-tuning pre-trained models on specific tasks, has been widely used to address limited data challenges. Existing literature highlights the importance of addressing domain differences between user and seller photos to improve matching accuracy.

**3. Problem Statement**:

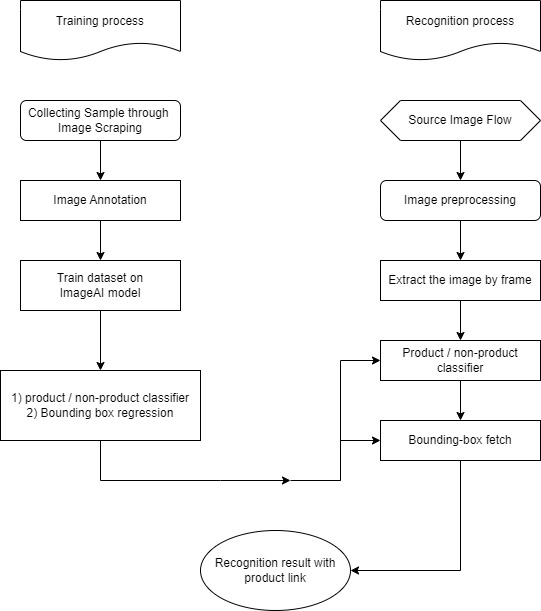
Recognizing products in user-generated images and accurately matching them with corresponding products in seller photos is a complex task with profound implications for e-commerce and retail. The presence of object bounding boxes in user photos acts as a guiding mechanism, aiding in the localization of desired products. This challenge necessitates the development of an advanced object detection model that not only identifies products within images but also facilitates seamless alignment and matching with the diverse array of products in the seller's inventory.

The challenge revolves around two distinct types of images: user photos, captured in real-world settings, and seller photos, carefully composed representations of products. The provided object bounding boxes specify desired products within user photos, serving as search queries. The project's objective is to develop an algorithm capable of accurately identifying and matching products in seller photos that correspond to the user-generated search queries.

**4. Design:**

Our solution hinges on the deployment of an object detection model built upon the YOLO (You Only Look Once) architecture. YOLO's distinct advantage lies in its ability to perform real-time object detection without compromising on accuracy. By selecting YOLOv3 and its more compact variant, TinyYOLOv3, we ensure a balance between detection speed and precision. The architecture's deep convolutional layers allow it to effectively capture intricate visual features, making it suitable for detecting products in various poses, lighting conditions, and orientations.

**Flow Chart**

****

**5. Description**

**5.1 Image Recognition vs. Image Detection**:

The terms image recognition and image detection are often used in place of each other. However, there are important technical differences. Image Detection is the task of taking an image as input and finding various objects within it. An example is face detection, where algorithms aim to find face patterns in images (see the example below). When we strictly deal with detection, we do not care whether the detected objects are significant in any way. The goal of image detection is only to distinguish one object from another to determine how many distinct entities are present within the picture. Thus, bounding boxes are drawn around each separate object. On the other hand, image recognition is the task of identifying the objects of interest within an image and recognizing which category or class they belong to.

**5.2 How does Image Recognition work?**

Using traditional Computer Vision The conventional computer vision approach to image recognition is a sequence (computer vision pipeline) of image filtering, image segmentation, feature extraction, and rule-based classification. However, engineering such pipelines requires deep expertise in image processing and computer vision, a lot of development time and testing, with manual parameter tweaking. In general, traditional computer vision and pixel-based image recognition systems are very limited when it comes to scalability or the ability to re-use them in varying scenarios/locations

Using Machine Learning and Deep Learning Image recognition with machine learning, on the other hand, uses algorithms to learn hidden knowledge from a dataset of good and bad samples (see supervised vs. unsupervised learning).

The introduction of deep learning, in combination with powerful AI hardware and GPUs, enabled great breakthroughs in the field of image recognition. With deep learning, image classification and face recognition algorithms achieve above-human-level performance and real-time object detection. Still, it is a challenge to balance performance and computing efficiency. Hardware and software with deep learning models have to be perfectly aligned in order to overcome costing problems of computer vision. Therefore, the ability to always use the most recent algorithm has direct costing implications: The most powerful and efficient algorithm requires several times cheaper hardware or achieves several times better performance on equivalent hardware when compared to legacy algorithms. Over the years, we have seen significant jumps in computer vision algorithm performance.

Compared to the traditional computer vision approach in early image processing 20 years ago, deep learning requires only engineering knowledge of a machine learning tool, not expertise in specific machine vision areas to create handcrafted features. While early methods required enormous amounts of training data, newer deep learning methods only need tens of learning samples. However, deep learning requires manual labeling of data to annotate good and bad samples, a process called image annotation. The process of learning from data that is labeled by humans is called supervised learning. The process of creating such labeled data to train AI models requires time-consuming human work, for example, to annotate standard traffic situations in autonomous driving.

The Process of Image Recognition Systems There are a few steps that are at the backbone of how image recognition systems work. Dataset with training data The image recognition models require training data (video, picture, photo, etc.). Neural networks need those training images from an acquired dataset to create perceptions of how certain classes look. For example, an image recognition model that detects different poses (pose estimation model) would need multiple instances of different human poses to understand what makes poses unique from each other. Training of Neural Networks for Image Recognition The images from the created dataset are fed into a neural network algorithm. This is the deep or machine learning aspect of creating an image recognition model. The training of an image recognition algorithm makes it possible for convolutional neural networks image recognition to identify specific classes. There are multiple well-tested frameworks that are widely used for these purposes today. AI Model Testing The trained model needs to be tested with images that are not part of the training dataset. This is used to determine the usability, performance, and accuracy of the model. Therefore, about 80-90% of the complete image dataset is used for model training, while the remaining data is reserved for model testing. The model performance is measured based on a set of parameters that indicate the percent confidence of accuracy per test image, incorrect identifications, and more. Read our article about how the evaluate the model performance in machine learning.

**6. Implementation:**

Data preprocessing will involve resizing images, normalizing pixel values, and extracting object bounding boxes. The model architecture will consist of multiple convolutional layers followed by fully connected layers. We will implement an attention mechanism that emphasizes the region specified by the bounding box. The model will be trained using a combination of user photos and corresponding seller photos.The core of our implementation is the utilization of the Detection Model training class. This class functions as an interface to streamline the training process of object detection models. To initiate the training, we preprocess the dataset comprising user photos along with their corresponding bounding box annotations following the YOLO annotation format. The training class then orchestrates the training of the YOLOv3 or TinyYOLOv3 model, incorporating the specifics of anchor boxes that are pivotal for detection accuracy.

**6.1 step involve**

**You Only Look Once (YOLO) :**YOLO stands for You Only Look Once, and true to its name, the algorithm processes a frame only once using a fixed grid size and then determines whether a grid box contains an image or not. For this purpose, the object detection algorithm uses a confidence metric and multiple bounding boxes within each grid box. However, it does not go into the complexities of multiple aspect ratios or feature maps, and thus, while this produces results faster, they may be somewhat less accurate than SSD. A very popular YOLO model is its third version, named YOLOv3; the latest and most powerfulversionisYOLOv7.

**How to apply Image Recognition Models Image Recognition** **with Python** :When it comes to image recognition, Python is the programming language of choice for most data scientists and computer vision engineers. It supports a huge number of libraries specifically designed for AI workflows – including image detection and recognition.

**Step #1:** To get your computer set up to perform python image recognition tasks, you need to download Python and install the packages needed to run image recognition jobs, including Keras.

**Step #2**: Keras is a high-level deep learning API for running AI applications. It runs on TensorFlow/Python and helps end-users deploy machine learning and AI applications using easy-to-understand code.

**Step #3**: If your machine does not have a graphics card, you can use free GPU instances online on Google Colab. For the purpose of classifying animals, there is a well-labeled dataset known as “Animals-10” that you can find on Kaggle. The dataset is totally free to download.

**Step #4**: Once you have obtained the online dataset from Kaggle by getting an API token, you can then start coding in Python after reuploading the necessary files to Google Drive. For more details on platform-specific implementations, several well-written articles on the internet take you step-by-step through the process of setting up an environment for AI on your machine or on your Collab that you can use.

**7 Architecture**

**YOLOv3 Architecture:**The YOLO (You Only Look Once) architecture, specifically YOLOv3, is a state-of-the-art deep learning model designed for real-time object detection. It divides the input image into a grid and assigns responsibility to each grid cell for predicting objects within its region. YOLOv3 stands out for its ability to detect a wide variety of objects in a single pass while maintaining good accuracy. Here's a detailed breakdown of its architecture:

**1. Input Processing:**

YOLOv3 takes an input image and divides it into a grid. Each grid cell is responsible for predicting objects within its region. The original image is resized to a fixed size, and the grid is typically 13x13, 26x26, or 52x52, depending on the model variant.

**2. Feature Extraction Backbone:**

The architecture uses a backbone network, such as Darknet-53, to extract high-level features from the input image. Darknet-53 consists of numerous convolutional layers that capture different levels of image information, from low-level edges to high-level object features.

**3. Detection at Multiple Scales:**

YOLOv3 employs a feature pyramid to capture objects at different scales. The network applies detection at three different scales, corresponding to the three grid sizes mentioned earlier (13x13, 26x26, 52x52). This enables the model to detect objects of various sizes in the input image.

**4. Anchor Boxes:**

YOLOv3 uses anchor boxes to predict object positions and sizes within each grid cell. These anchor boxes are pre-defined shapes that the model learns to adjust to fit the objects in the image. Different anchor boxes are responsible for detecting objects of different scales.

**5. Prediction at Each Scale:**

For each grid cell in each scale, YOLOv3 predicts bounding boxes, objectness scores, and class probabilities. Each bounding box is responsible for predicting the coordinates of a detected object's bounding box, a confidence score indicating the presence of an object, and class probabilities for different object categories.

**6. Non-Maximum Suppression (NMS):**

After predictions are made, a post-processing step called Non-Maximum Suppression (NMS) is applied to remove duplicate and overlapping bounding box predictions. NMS helps to retain only the most confident and accurate detections

**7.1 Integration with our Project:**

In the context of your project, YOLOv3 serves as a powerful tool for detecting products in user photos and aligning them with corresponding products in seller photos. The YOLOv3 model would be fine-tuned using the provided object bounding box annotations to learn the specific characteristics of the products you're interested in.

During the training process, you would use the YOLOv3 architecture and update the anchor boxes and class probabilities according to the product categories you're dealing with. The model would learn to identify the products in user-generated images and subsequently match them with the seller's product gallery based on the detected bounding boxes.

The YOLOv3 architecture, with its multi-scale detection and anchor box mechanisms, provides a robust framework for solving the complex problem of product recognition and matching that your project entails.

**8. Training and Implementation**

To get started, you need prepare your dataset in the **YOLO annotation format** and organize it as detailed below:

– Decide the type of object(s) you want to detect and collect about 200 (minimum recommendation) or more picture of each of the object(s)

– Once you have collected the images, you need to annotate the object(s) in the images. You can use a tool like [Label IMG](https://github.com/tzutalin/labelImg) to generate the annotations for your images.

– Once you have the annotations for all your images, create a folder for your dataset **(E.g headsets)** and in this parent folder, create child folders **train** and **validation**

– In the **train** folder, create **images** and **annotations** sub-folders. Put about 70-80% of your dataset of each object’s images in the **images** folder and put the corresponding annotations for these images in the **annotations** folder.

– In the **validation** folder, create **images** and **annotations** sub-folders. Put the rest of your dataset images in the **images** folder and put the corresponding annotations for these images in the **annotations** folder.

**1 Installing Required Packages:**

!pip install cython pillow>=7.0.0 numpy>=1.18.1 opencv-python>=4.1.2 torch>=1.9.0 --extra-index-url https://download.pytorch.org/whl/cpu torchvision>=0.10.0 --extra-index-url https://download.pytorch.org/whl/cpu pytest==7.1.3 tqdm==4.64.1 scipy>=1.7.3 matplotlib>=3.4.3 mock==4.0.3

This command installs various Python packages required for your project, including packages for deep learning (PyTorch), computer vision (OpenCV), and other utilities (pytest, tqdm, etc.). It also specifies the versions of these packages to ensure compatibility.

**2. Mounting Google Drive:**

from google.colab import drive

drive.mount('/content/drive')

This code snippet connects Google Drive to your Colab environment, enabling you to access files and data stored in your Google Drive

**3. Installing ImageAI Library:**

pip install imageai --upgrade

This command installs the ImageAI library, which provides tools for working with image recognition and object detection tasks.

**4. Downloading Pretrained YOLOv3 and TinyYOLOv3 Models:**

!wget https://github.com/OlafenwaMoses/ImageAI/releases/download/3.0.0-pretrained/yolov3.pt

!wget https://github.com/OlafenwaMoses/ImageAI/releases/download/3.0.0-pretrained/tiny-yolov3.pt

These commands download the pretrained YOLOv3 and TinyYOLOv3 models. These models are already trained on a large dataset and can be fine-tuned for your specific task.

**5. Importing Required Modules and Creating the Trainer:**

from imageai.Detection.Custom import DetectionModelTrainer

trainer = DetectionModelTrainer()

These lines import the necessary modules from the ImageAI library and create an instance of the `DetectionModelTrainer` class, which is used to train custom detection models.

**6. Setting Model Type and Configuration**

trainer.setModelTypeAsYOLOv3()

trainer.setDataDirectory(data\_directory="/content/drive/MyDrive/SplitData/SplitData")

Here, you set the model type to YOLOv3 and specify the data directory where your dataset is located. Make sure to replace `"/content/drive/MyDrive/SplitData/SplitData"` with the actual path to your dataset.

**7. Configuring Training:**

trainer.setTrainConfig(object\_names\_array=["Aapple Watch Series 8", "Casio G-Shock", "Casio Vintage ( A-158WA-1Q ) Digital Watch 2", "Fastrack Minimalists Analog Watch", "Fire-Boltt Ninja Calling Pro Plus", "Fossil Briggs Analog - CH2927I","Noise Evolve 3","One PLus - Band","Samsung Watch 4"], batch\_size=4, num\_experiments=51, train\_from\_pretrained\_model="/content/yolov3.pt")

In this part, you configure the training settings. You specify the names of the object categories you want to detect (replace the example names with your actual product categories). You set the batch size for training, the number of training experiments, and the path to the pretrained YOLOv3 model.

**8. Initiating Training:**

trainer.trainModel()

Finally, you initiate the training process using the `trainModel()` method. This will start training your YOLOv3-based custom object detection model on your dataset.

Training a custom object detection model is a resource-intensive task that may take some time to complete. Ensure you have enough computational resources and time to execute the training process.

**9. Results and Evaluation**:

We approach the evaluation from two angles. First, we assess the object detection accuracy. This is achieved by employing metrics such as Intersection over Union (IoU) to measure the degree of overlap between predicted and ground truth bounding boxes. Second, the product matching accuracy is gauged by quantifying precision, recall, and F1-score. This holistic evaluation strategy accounts for both the ability to accurately locate products within images and the proficiency in matching these products across user and seller images.

The performance of the model will be evaluated based on accuracy, precision, recall, and F1-score. We will create a test set comprising user-generated queries and their corresponding seller images. Confusion matrices and visualizations will help analyze the model's ability to match products accurately.

**10. Future Scope:**

This project has potential for further enhancements. Future iterations could explore the integration of domain adaptation techniques to better align user and seller photo representations. Real-time matching, scalability, and optimization for larger galleries could also be considered. Additionally, user feedback could be used to refine and improve the algorithm's performance over time.

The inclusion of the object detection model opens avenues for future enhancements. A subsequent phase could concentrate on optimizing the model's trade-off between accuracy and real-time detection speed. Exploring alternative object detection architectures, such as Faster R-CNN or SSD, could uncover avenues to further improve detection performance. Additionally, a more comprehensive system could fuse object detection with image classification, refining the matching process through a multi-modal approach.

**11. Conclusion:**

In conclusion, this project tackles the challenge of recognizing products in user photos and accurately matching them with products in seller photos. By employing a combination of deep learning techniques, attention mechanisms, and transfer learning, we aim to bridge the gap between user-generated content and seller representations, thereby enhancing the accuracy and efficiency of product matching. our project embarks on the intricate journey of elevating product recognition and matching. The application of YOLOv3 and TinyYOLOv3 in object detection serves as a cornerstone to accurately identify products within user images and facilitate seamless product matching against seller photos. This innovative approach endeavors to enhance the shopping experience by bolstering accuracy and relevance in the product discovery process.