INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 4 - V7I4-1656)
Available online at: https://www.ijariit.com

Identification of Jewelry Article using Transfer Learning and Image Repository

Mohammed Mafaz
mafazazhaar@gmail.com
Loyola College, Chennai, Tamil Nadu

ABSTRACT

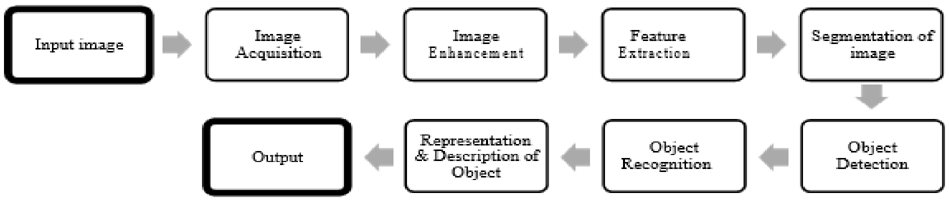
"IDENTIFICATION of jewelry ARTICLE USING TRANSFER LEARNING AND IMAGE REPOSITORY" may be a project that aims to automate, boost and revolutionize the task of recognizing Jewellery articles for tagging before adding the article into the inventory. the aim of this project is to optimize the task of stock entry and to scale back the manual work of tagging each and each article. The project involves creating image repository of varied Jewellery articles by manually clicking the pictures , label the pictures supported the classes, train the model and eventually to classify the articles in real-time using live camera feed using Transfer Learning, and OpenCV. Once the article is recognized, the model will pass the worth vectors including name of the article, image of the article and number of pieces of a specific article skilled the live camera. this complete process will help in boosting up the manual process of adding Jewellery article to inventory, will reduce manpower, less erroneous and can economize for the stakeholders during a end of the day.

Keywords: *Transfer learning, YOLO, Darknet, LabelImg, Computer Vison.*

1. INTRODUCTION

Object detection and tracking is one among the critical areas of knowledge science world thanks to routine change in motion of object and variation in scene size, occlusions, appearance variations, and ego-motion and illumination changes. Specifically, feature selection is that the vital role in object tracking. it's associated with many real time applications like vehicle perception, video surveillance then on. so as to beat the difficulty of detection, tracking associated with object movement and appearance. Most of the algorithm focuses on the tracking algorithm to smoothen the video sequence. On the opposite hand, few methods use the prior available information about object shape, color, texture then on. Tracking algorithm which mixes above stated parameters of objects is discussed and analyzed during this research.

All visual perception has two parts- Category Recognition and its detection. Category Detection deals with distinguishing the thing from the background. In addition, Category Recognition deals with classifying the thing into one among the predefined categories. It's an identifying process of specific object during a digital image or video. Generally, visual perception algorithms believe matching, learning, or pattern recognition algorithms using appearance-based or feature-based techniques. for instance , it's wont to find instances of real-life objects like bicycles, fruits, animals and buildings in images or videos. Object detection algorithms use features , which may be extracted to acknowledge a specific object. This model is extremely simple and straightforward to implement. Here, object detection may be a single regression problem, which detects directly from bounding box coordinates and sophistication probability. Every object has its own class like all circles are round, which are used while recognizing the objects. Object detection follows a series of steps:



Advanced concepts like neural networks and deep learning are gaining its ground within the area of computer vision. the answer provided using these techniques are often highly adaptive and reliable in real time.

A smart system is required to form sure safety and to make the individual highly conscious of his/her surrounding to enhance assistance. Before implementing object detection and classifying the thing supported its category, we'd like to know the difference between object detection and image classification. Image classification is that the process of classifying the image to a category supported the recognized features and patterns, whereas object detection is that the process of obtaining the bounding box of coordinates exactly where a specific object is present within the image. we will detect quite one object of a special class in a picture . In short, object detection can't only tell us what's in a picture but also where the thing is also . There are several ways to detect objects in a picture.

YOLO may be a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, YOLO frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. one neural network predicts bounding boxes and sophistication probabilities directly from full images in one evaluation. Since the entire detection pipeline may be a single network, it are often optimized end-to-end directly on detection performance. Our unified architecture is extremely fast. Base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the MAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is a smaller amount likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Transfer learning is that the improvement of learning during a new task through the transfer of data from a related task that has already been learned. While most machine learning algorithms are designed to deal with single tasks, the event of algorithms that facilitate transfer learning may be a topic of ongoing interest within the machine-learning community. Every object class has its own special features that helps in classifying the category – for instance all circles are round. Object class detection uses these special features. for instance , when trying to find circles, objects that are at a specific distance from some extent (i.e. the center) are sought. Similarly, when trying to find squares, objects that are perpendicular at corners and have equal side lengths are needed. an identical approach is employed for face identification where eyes, nose, and lips are often found and features like complexion and distance between eyes are often found.

Typically, there are three steps in an object detection framework.

(a) First, a model or algorithm is used to generate regions of interest or region proposals. These region proposals are a large set of bounding boxes spanning the full image (that is, an object localization component).
(b) In the second step, visual features are extracted for each of the bounding boxes, they are evaluated and it is determined whether and which objects are present in the proposals based on visual features (i.e. an object classification component).
(c) In the final post-processing step, overlapping boxes are combined into a single bounding box (that is, non-maximum suppression).

Some of the tools and technologies used for object detection are:

1.1 LabelImg:

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO format. The text file with the class and coordinates of respective image will be saved when the YOLO option is selected.

1.2 YOLO

YOLO stands for “You Only Look Once”, is a state-of-the-art algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in **one run of the Algorithm.** YOLO doesn’t search for interested regions in the input image that could contain an object, instead it splits the image into cells, typically 19x19 grid. Each cell is then responsible for predicting K bounding boxes. An Object is considered to lie in a specific cell only if the center co-ordinates of the anchor box lie in that cell. Due to this property the center co-ordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole Image size.

1.3 Darknet

Darknet is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. Darknet works side by side with YOLO to make the model more robust and accurate.

1.4 Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

1.5 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

2. METHODOLOGY

2.1 About the dataset

The dataset (i.e., image repository) used for training and testing the model consist of 247 images including manually clicked images of Jewellery article using camera and some images from google just to make the model more robust. The image repository consists of images of 5 different jewellery article. The article's images used to train the model are:

- Ring.
- Chains.
- Bangles.
- Bracelets.
- Ear rings.

Each and every image in the dataset will have a text file, that consists of number of classes and the coordinates at which the jewellery article is present in the image. This text file is used by the algorithm to locate the jewellery article in the same image as the text file.

2.2 Data Source

The jewellery article used for clicking the images was provided by the local jewellery vendor form Chennai. The jewellery vendor is a client of AUC ventures and they were very generous for letting us click the images.

2.3 Tools

- Google Colab.
- Jupyter Notebook.
- LabelImg.
- Drive.
- Microsoft Office.

Figure 4: Tools

Language used: Python

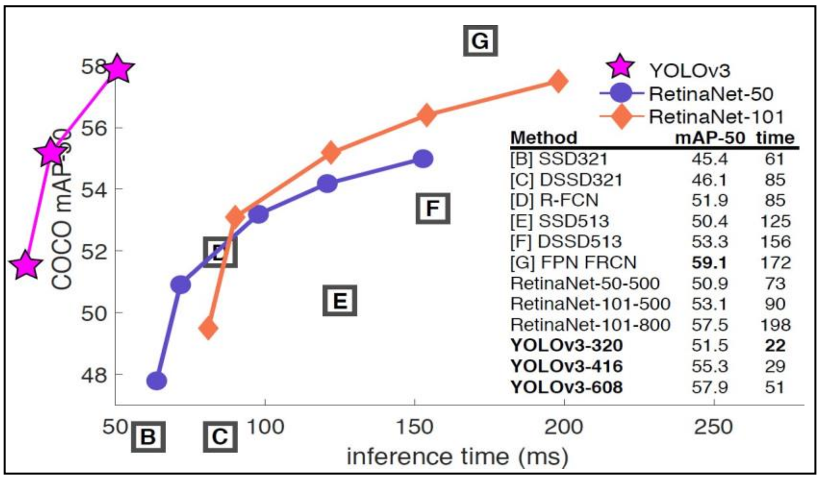
Google Colab was used to train the model because Google Colab provides free GPU which is helpful in faster training of the model.

Jupyter Notebook was used to test the trained model with the help of OpenCV.

Google drive worked as a storage drive to Google Colab. It contained all the training and testing images. Once the model was trained the model itself and the save-points were saved in google drive.

2.3.1 YOLOv3:

YOLO stands for "You Only Look Once", is a state-of-the-art algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm. YOLO doesn't search for interested regions in the input image that could contain an object, instead it splits the image into cells, typically 19x19 grid. Each cell is then responsible for predicting K bounding boxes. An Object is considered to lie in a specific cell only if the center co-ordinates of the anchor box lie in that cell. Due to this property the center co-ordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole Image size. Yolov3 is extremely powerful algorithm for object detection, it can detect objects with mAP of 57.9 within 51 milli second on COCO Dataset. (COCO Dataset consist of thousands of images of 80 different classes).



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 5: Object detection algorithms accuracy graph

From figure 5 it is clearly inferred that all the YOLOv3 have higher accuracy and lower computational time compared to other models of same family.

The YOLO models are end-to-end deep learning models and are well-liked because of their detection speed and accuracy (figure 4). Additionally, the methods learn generalizable representations of objects which is of essence when a model is applied in real life. The structure of a YOLO network is similar to a normal CNN. It consists of several convolutional and max pooling layers, ending with two fully connected layers.

Previous methods, like region-based convolutional neural networks (R-CNN), require thousands of network evaluations to make predictions for one image which can be time- consuming and painful to optimize. YOLO focuses on a specific area of the image and trains each individual component separately. A YOLO model on the other hand, only passes the image once through the neural network ("You Only Look Once").

The network divides the image into a grid of cells, which all predict five bounding boxes and object classifications. The boxes having a low probability of containing an object, and the ones sharing large areas with other boxes will be removed by a process called non-maximal suppression.

The most current of the main versions is the third iteration of the approach, namely YOLOv3. In each version improvements have been made over the previous one. The initial version proposed the general architecture, after which the second variation improved accuracy significantly while making it faster. YOLOv3 refined the design further by using tricks, such as multi-scale prediction and bounding box prediction through the use of logistic regression. While the accuracy increased dramatically with this version, it traded off against speed which reduced from 45 to 30 frames per second.

YOLOv3 uses a variant of Darknet, a framework to train neural networks, which originally has 53 layers. For the detection task another 53 layers are stacked onto it, accumulating to a total of a 106-layer fully convolutional architecture. This explains the reduction in speed in comparison with the second version, which only has 30 layers.

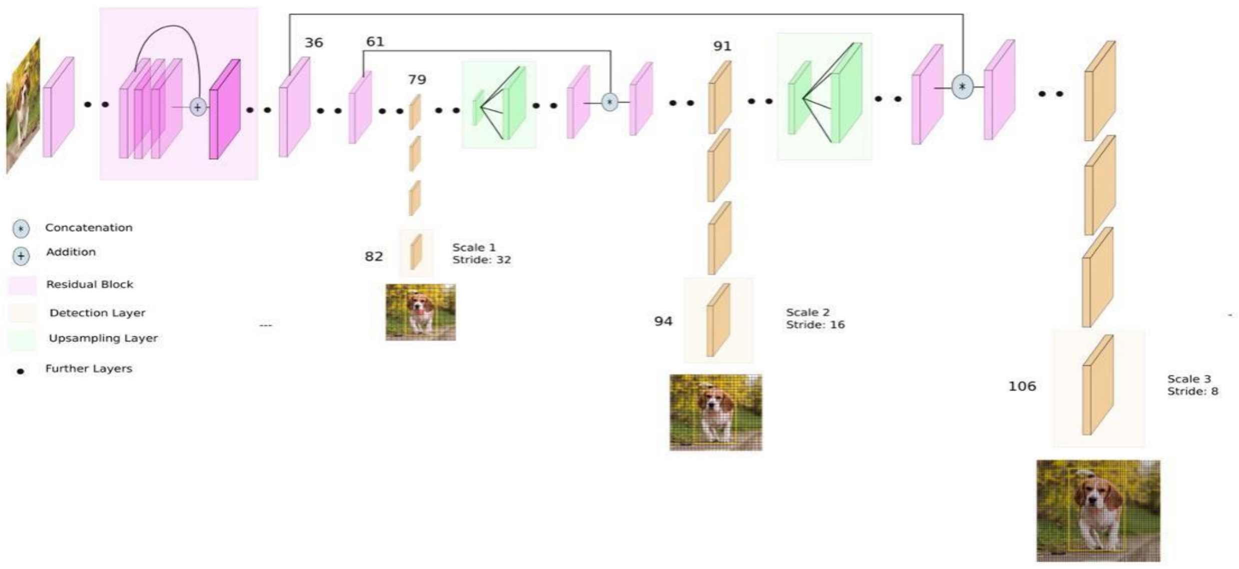


Figure 6: YOLO Architecture

In the convolutional layers, kernels of shape 1x1 are applied on feature maps of three different sizes at three different places in the network. The algorithm makes predictions at three scales, given by downsampling the dimensions of the image by a stride of 32, 16, 8 respectively. Downsampling, the reduction in spatial resolution while keeping the same image representation, is done to reduce the size of the data. Every scale uses three anchor bounding boxes per layer. The three largest boxes for the first scale, three medium ones for the second scale and the three smallest for the last scale. This way each layer excels in detecting large, medium or small objects.
Previously in YOLO, the softmax activation function determining the classes of objects in bounding boxes was also employed in these models. Authors of YOLOv3 have refrained from softmaxing the classes, since the method rests on the assumption that classes are mutually exclusive. For example, if there are classes like "cat" and "animal" in the dataset and one of the objects in the bounding boxes is a cat, this assumption fails, because a cat is also an animal. Instead, independent logistic classifiers predict each class score and a threshold is used to perform multilabel classification for objects detected in images. An element belonging to a certain class will not be influenced by the decision of that element belonging to another class (binary cross-entropy loss).

2.3.2 LabelImg:

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO format. The text file with the class and coordinates of respective image will be saved when the YOLO option is selected.

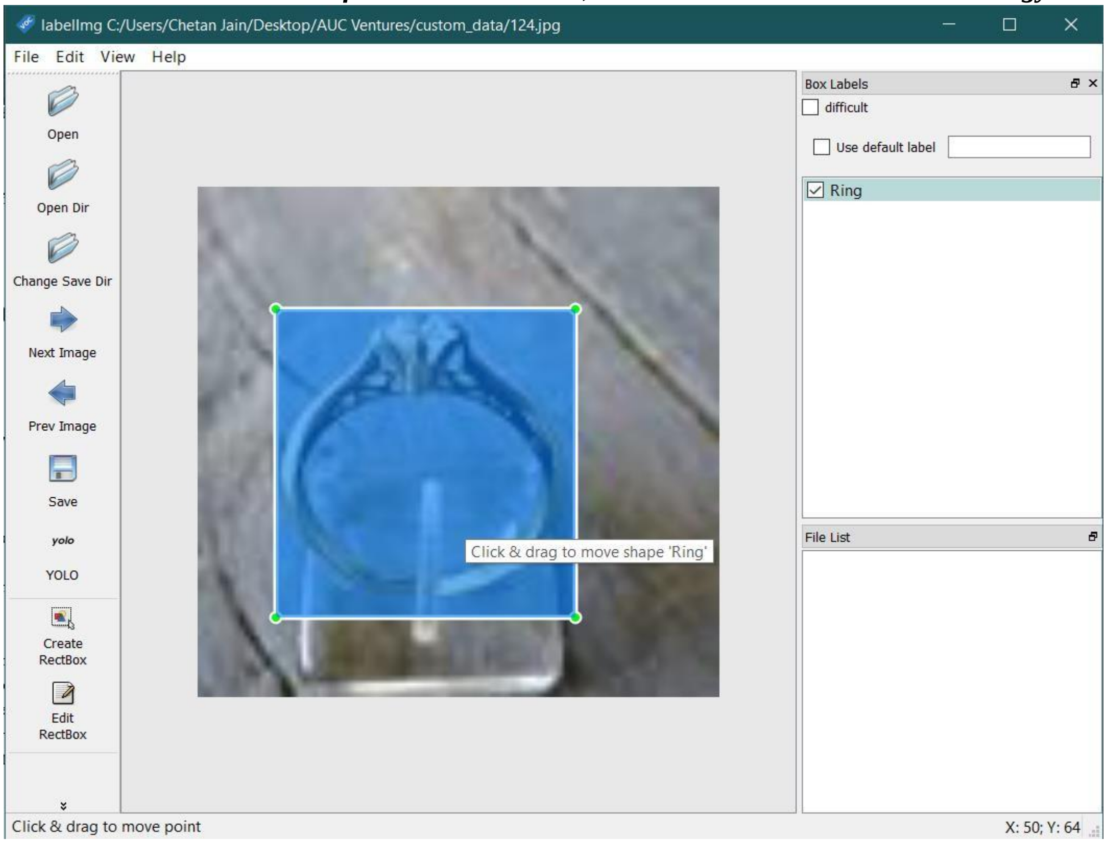
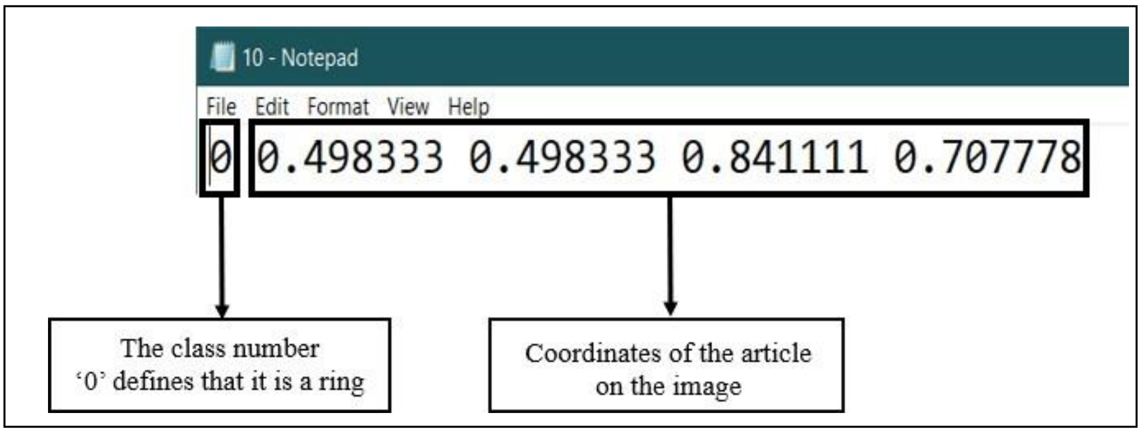


Figure 7: LabelImg Image mapping on objects

Figure 7 displays the screenshot of LabelImg application. Using mouse, a rectangle can be mapped around the object to make the text file in order to pass it on to the algorithm.


10 - Notepad
File Edit Format View Help
The class number '0' defines that it is a ring
Coordinates of the article on the image

```
0 0.498333 0.498333 0.841111 0.707778
```

Figure 8: Sample txt file after labelling

In figure 8, The first '0' in the figure defines what is the class of the article in the image, rest 4 values which are separated by a single space are the coordinates of the article (object) in the image, in which the first value is x value, second is y value, third is width and fourth is the height.

2.3.3 Darknet:

Darknet is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. Darknet works side by side with YOLO to make the model more robust and accurate. Darknet uses C library's as in a lot of algorithms are written in C. There are a few different implementations of the YOLO algorithm on the web. Darknet is one such open-source neural network framework. Darknet is written in the C Language and CUDA technology, which makes it really fast and provides for making computations on a GPU, which is essential for real-time predictions.

2.3.4 Transfer Learning:

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

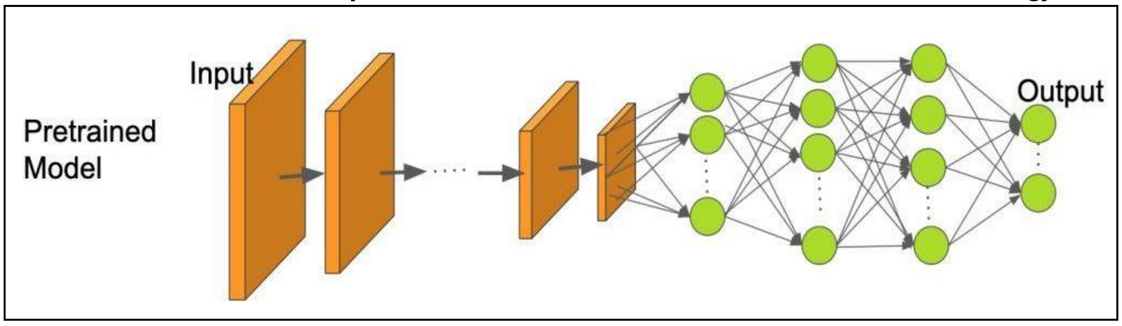


Figure 9: Pre-Trained Model

Figure 9 is the representation of a pre-trained model. The Fully-connected part of the model is what exactly being tuned manually based on the dataset to make a model which is already learned before. Once the Fully-connected part is tuned, the model is ready to be trained on the custom dataset

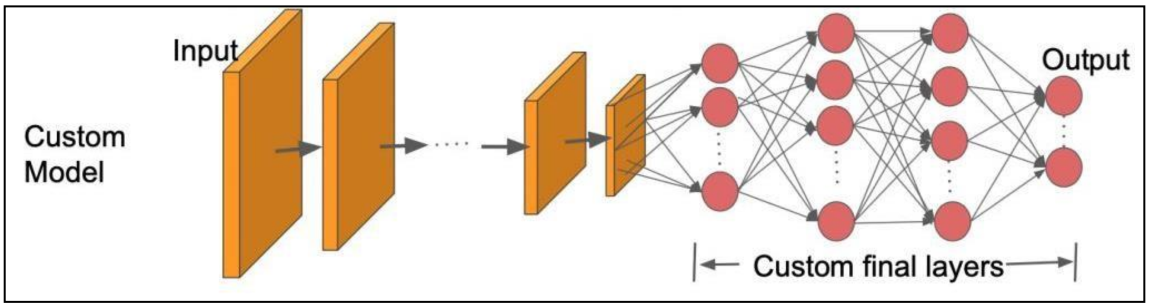


Figure 10: Custom Pre-Trained Model

Figure 10 displays the customized pre-trained model which is using the same weights as of before and will be extremely fast and accurate because the model is using the pre- trained weights.

2.3.5 OpenCV:

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. OpenCV deals with modeling and replicating human vision using computer software and hardware.

2.3.6 Loss v/s Iteration curve

The loss v/s iteration curve is a line plot with loss on Y axis and iteration (epochs) on X axis. The loss v/s Iteration curve shows how the loss is increasing or decreasing with respect to the iteration. If the curve shows a nice fall and a flat line towards left that means the loss is decreasing and model is getting trained accurately.

2.3.7 Precision

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives. False positives are the cases that the model incorrectly labels as positive that are actually negative.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Figure 11: Precision Formula

2.3.8 Recall

Recall is the number of true positives divided by the number of true positives plus the number of false negatives. True positives are data point classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

Figure 12: Recall Formula

2.3.9 F1-Score

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively F1 is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if the data have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1 - Score = 2 \times \frac{Precision \cdot Recall}{Precision + Recall}$$

Figure 13: F1-Score Formula

2.3.10 Intersection over Union (IoU)

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. Intersection over Union is simply an evaluation metric. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.

$$IoU = \frac{Area\ Of\ Overlap}{Area\ of\ Union}$$

Figure 14: IoU Formula

2.3.11 Mean Average Precision (mAP)

mAP (mean average precision) is the average of Average Precision. In some context, mAP is computed by the Average Precision for each class and average them.

2.4 FLOWCHART

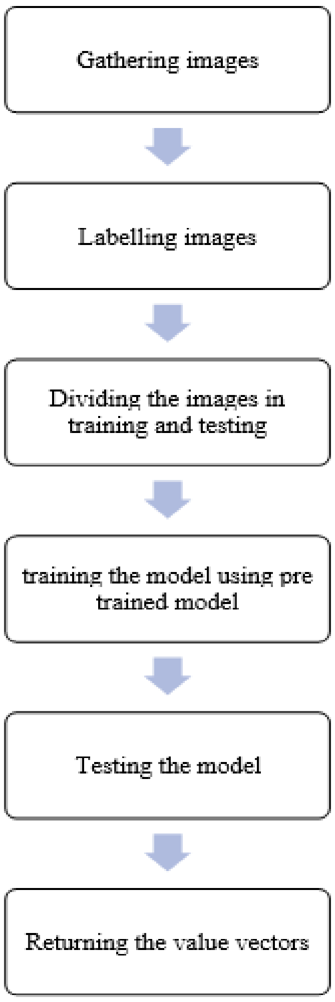
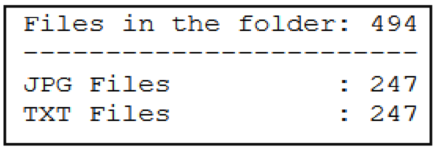

Gathering images
Labelling images
Dividing the images in training and testing
training the model using pre trained model
Testing the model
Returning the value vectors

Figure 15: Flowchart

5. ANALYSIS AND INTERPRETATION

5.1 DATA EXPLORATION



```
Files in the folder: 494
------------------------
JPG Files          : 247
TXT Files          : 247
```

Figure 16: Contents of the folder

The dataset used for training the model has 494 files including 247 images and 247 annotated text files.

5.2 LOSS V/S ITERATION CURVE

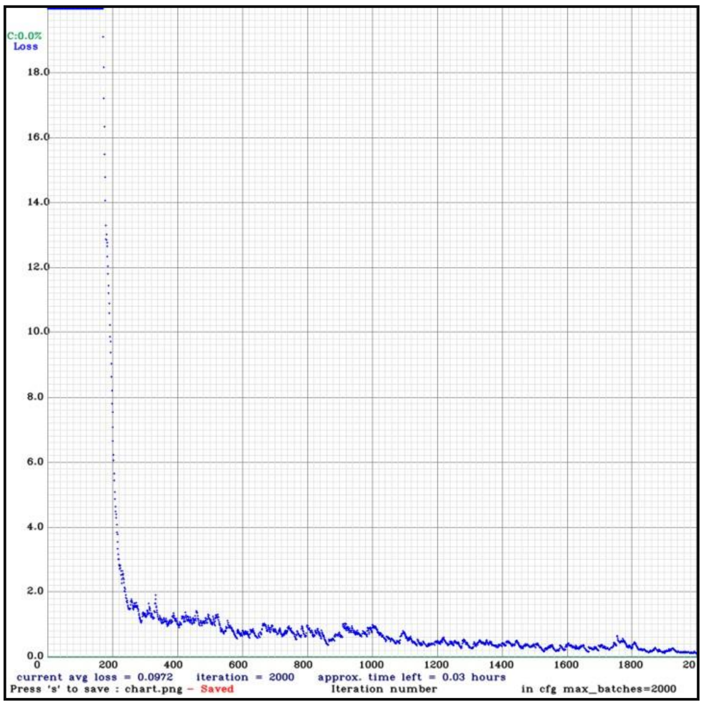
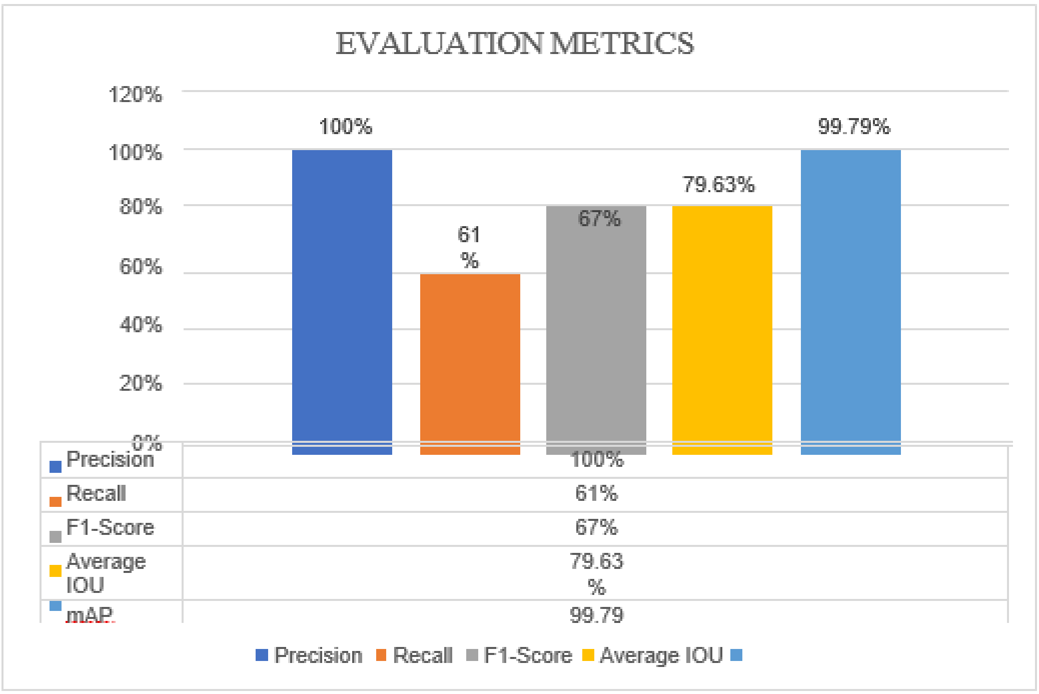


Figure 17: Loss V/S Iteration curve

From figure 17, It is inferred that the first 180 (approx.) iteration (Epochs) has maximum loss and after that the loss starts to drop. From 180th iteration to 220th iteration (Epoch), the loss drops from almost 100% to 20%. Loss, from there constantly has some highs and lows between 20% to 5%. At 2000th iteration the loss is minimum, somewhere around 1% to 0%. The average loss for all the 2000 iterations is 0.0972. Since the loss is extremely low, the model is optimal for further use.

5.3 EVALUATION METRICS



Precision	100%
Recall	61%
F1-Score	67%
Average IOU	79.63 %
mAP	99.79

Figure 18: Evaluation Matrices

5.3.1 PRECISION

Precision attempts to answer the question that what proportion of positive identifications was actually correct. Form figure 5 it is inferred that the precision of model is 1.0 i.e., 100%. That means that the model predicted no false positive value.

5.3.2 RECALL

Recall attempts to answer the question that what proportion of actual positives was identified correctly. From figure 5 it is inferred that the recall of model is 0.61 i.e., 61%. That means that the model has some false negative values.

5.3.3 F1-SCORE

F1-score is the harmonic mean of precision and recall. From the figure 5 it is inferred that the F1-score is 67%.

5.3.4 AVERAGE INTERSECT OVER UNION (IOU)

Optimum IOU value need to be greater than 0.5 or 50%. YOLO sets the threshold as 0.5, any IOU greater than 0.5 is considered as a good IOU value. The average IOU value for the model is 0.7963 or 79.63%, which is more than the threshold and is considered as optimal value.

5.3.5 MEAN AVERAGE PRECISION (mAP)

The Mean Average Precision (mAP) value is considered as the most important parameter to judge the accuracy of any object detection model. From figure 5 it is inferred that mAP value of the model is 99.79%. Greater the mAP, better the model.

Table 2: Overall Metrics of Evaluation

Overall Metrics of evaluation			
No.	**Matric name**	**Threshold value**	**Obtained Value**
1	Precision	> 0.7	1.0
2	Recall	> 0.7	0.61
3	F1-score	> 0.6	0.67
4	IOU	> 0.5	0.7963
5	mAP	Greater the better	0.9979

3. CONCLUSION

- Object detection is breaking into a wide range of industries, with use cases ranging from personal security to productivity in the workplace. Facial detection is one form of it, which can be utilized as a security measure to let only certain people into a highly classified area of a government building, for example. Object detection can be used to count the number of people present within a business meeting to automatically adjust other technical tools that will help streamline the time dedicated to that particular meeting. It can also be used within a visual search engine to help consumers find a specific item they're on the hunt for.
- The future of object detection has massive potential across a wide range of industries. one of them is jewellery industry.
- Jewellery industry is one of the riskiest business in terms of value, money and overall risk of fraud. In these kinds of critical scenario's, projects like "Identification of jewellery article using transfer learning and image repository" helps vendors in a lot of ways such as saving time while making a stock entry, saving money as this project aim to reduce human intervention in the process, reducing the risk of man-made errors and most importantly this project will save time which is extremely crucial in this industry.
- One of the biggest risks associated with inventory control in jewellery industry is theft, especially when it comes to high-value inventory stock. Companies spend millions of dollars annually on security measures to safeguard inventory and prevent theft, however it still occurs on a regular basis.
- There are a lot of checks a jewellery article needs to pass before getting included in the inventory of the firms. Checks such as, weight, purity etc. are taken into consideration. The above-mentioned process is extremely tedious and time consuming.
- By making use of Artificial intelligence and deep learning in making projects like this, will add a lot of values not only in terms of speeding up the process but also in monetary terms in business for a longer run.
- Project like this are inexpensive and require little or no maintenancesomething like a one-time investment.
- With regular enhancement in AI and deep learning, algorithms used for modelling can be tuned for better accuracy over a period of time. So, there is no chance that projects like these will go out of the fashion.
- Coupling of deep learning algorithms with other frameworks such as OpenCV are so simplified that if in future the proposed algorithm is outdated, with just some minor changes, it is possible to remove the outdated algorithm and embed a new and fresh yet more powerful algorithm in place.
- YOLOv3, algorithm which is used to implement this project is highly accurate and extremely fast. This algorithm can be deployed on low processing hardware's such as raspberry PI's.
- By making use of IoT devices such as Arduino, Raspberry PI and Deep learning, one can solve a lot of real-life problems.

REFERENCES

[1] Ajeet Ram Pathaka, Manjusha Pandeya, Siddharth Rautaraya, (2018), "Application of Deep Learning for Object Detection", "ICCIDS"

[2] Akansha Bathija, Prof. Grishma Sharma, (2019), "Visual Object Detection and Tracking using: YOLO and SORT", "IJERT"

[3] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, (2017), "ImageNet Classification with Deep Convolutional Neural Networks", "University of Toronto press"

[4] Behnam Neyshabur, Hanie Sedghi, Chiyuan Zhang, (2020), "What is being transferred in transfer learnning?", "Google"

[5] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu, (2018), "A Survey on Deep Transfer Learning", "IEEE"

[6] Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, (2014), "How transferable are features in deep neural networks?","Cornell University Press"

[7] Jiuxiang Gua, Zhenhua Wangb, Jason Kuenb, Lianyang Mab, Amir Shahroudyb, Bing Shuaib, Ting Liub, Xingxing Wangb,

Li Wangb, Gang Wangb, Jianfei Caic, Tsuhan Chenc, (2017), "Recent Advances in Convolutional Neural Networks", "arXiv:1512.07108v6"
[8] Joseph Redmon, Ali Farhadi, (2018), "YOLOv3: An Incremental Improvement", "arXiv:1804.02767v1",
[9] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, (2016), "You Only Look Once: Unified, Real-Time Object Detection", "IEEE"
[10] Junjie Yan, Yinan Yu, Xiangyu Zhu, Zhen Lei, Stan Z. Li, (2015), "Object Detection by Labeling Superpixels", "IEEE"
[11] Kai Kang, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, (2016), "Object Detection from Video Tubelets with Convolutional Neural Networks", "IEEE"
[12] Karl Weiss, Taghi M. Khoshgoftaar, DingDing Wang, (2016), "DOI 10.1186/s40537-016-0043-6 Journal of Big Data",
[13] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, Matti Pietikäinen, (2020), "Deep Learning for Generic Object Detection: A Survey", "IJCV"
[14] Lisa Torrey, Jude Shavlik, (2009), "Transfer Learning", "IGI Global"
[15] Niklas Fiedler, Marc Bestmann, N. Hendrich, (2018), "ImageTagger: An Open Source Online Platform for Collaborative Image Labeling", "ResearchGate"
[16] P.Devaki, S.Shivavarsha, G.Bala Kowsalya, M.Manjupavithraa, E.A. Vima, (2019), "Real-Time Object Detection using Deep Learning and Open CV", "IJITEE"
[17] Peilin Zhao, Steven C.H. Hoi, (2010), "OTL: A Framework of Online Transfer Learning","IEEE"
[18] Sandeep Kumar, Aman Balyan, Manvi Chawla, (2017), "Object Detection and Recognition in Images", "IJEDR"
[19] Sushma L, Dr. K.P. Lakshmi,(2020), "An Analysis of Convolution Neural Network for Image Classification using Different Models", "IJERT"

APPENDIX

SOURCE CODE

- Model Training

```
from google.colab import drive drive.mount('/content/drive')
!ls '/content/drive/My Drive/yolo_custom_model_Training'
!unzip '/content/drive/My Drive/yolo_custom_model_Training/custom_data.zip' -d '/content/drive/My Drive/yolo_custom_model_Training/'
!git clone 'https://github.com/AlexeyAB/darknet.git' '/content/drive/My Drive/yolo_custom_model_Training/darknet'
%cd '/content/drive/My Drive/yolo_custom_model_Training/darknet'
!make
%cd '/content/drive/My Drive/yolo_custom_model_Training'
!chmod +x ./darknet/darknet
!darknet/darknet
%cd '/content/drive/My Drive/yolo_custom_model_Training'
!python custom_data/creating-files-data-and-name.py
!python custom_data/creating-train-and-test-txt-files.py
!darknet/darknet detector train custom_data/labelled_data.data darknet/cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74
'/content/drive/My Drive/yolo_custom_model_Training/custom_data/10.jpg' -thresh 0.3
```

-Model Testing

```
import cv2
import numpy as np
import matplotlib.pyplot as plt net =
cv2.dnn.readNetFromDarknet("yolov3_custom.cfg",r"Downloads\yolov3_cus tom_2000.weights")
classes = ['Ring', 'Chains', 'Bangles', 'Bracelets', 'Ear rings'] cap = cv2.VideoCapture(0)

while 1:
_, img = cap.read()
img = cv2.resize(img,(1280,720)) hight,width,_ = img.shape
blob = cv2.dnn.blobFromImage(img, 1/255,(416,416),(0,0,0),swapRB
= True,crop= False) net.setInput(blob)
output_layers_name = net.getUnconnectedOutLayersNames() layerOutputs = net.forward(output_layers_name)
boxes =[] confidences = [] class_ids = []

for output in layerOutputs: for detection in output:
score = detection[5:] class_id = np.argmax(score) confidence = score[class_id] if confidence > 0.7:
center_x = int(detection[0] * width) center_y = int(detection[1] * hight) w = int(detection[2] * width)
h = int(detection[3]* hight) x = int(center_x - w/2)
y = int(center_y - h/2) boxes.append([x,y,w,h]) confidences.append((float(confidence))) class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes,confidences,.5,.4) boxes =[]
confidences = [] class_ids = []

for output in layerOutputs: for detection in output:
score = detection[5:] class_id = np.argmax(score) confidence = score[class_id] if confidence > 0.5:
```

```
center_x = int(detection[0] * width) center_y = int(detection[1] * hight)
 w = int(detection[2] * width) h = int(detection[3]* hight)

x = int(center_x - w/2) y = int(center_y - h/2)


boxes.append([x,y,w,h]) confidences.append((float(confidence))) class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes,confidences,.8,.4) font = cv2.FONT_HERSHEY_PLAIN
colors = np.random.uniform(0,255,size =(len(boxes),3)) if       len(indexes)>0:
for i in indexes.flatten(): x,y,w,h = boxes[i]
label = str(classes[class_ids[i]]) confidence = str(round(confidences[i],2)) color = colors[i]
cv2.rectangle(img,(x,y),(x+w,y+h),color,2) cv2.putText(img,label + " " + confidence,
(x,y+400),font,2,color,2)

cv2.imshow('img',img)
if cv2.waitKey(1) == ord('q'): break

cap.release() cv2.destroyAllWindows()
```