Visvesvaraya Technological University, Belagavi.

PROJECT REPORT

on

"ID CARD DETECTION AND FACIAL RECOGNITION USING OPENCV AND TENSORFLOW"

Project Report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

For the academic year 2019-20

Submitted by

1CR16EC031 CHARAN KUMAR M J

1CR16EC067 KUSHAL KUMAR B V

1CR16EC068 KUSHAL M

Under the guidance of

Prof. Pappa M

Associate Professor

Department of ECE

CMRIT, Bengaluru

Department of Electronics and Communication Engineering

CMR Institute of Technology, Bengaluru – 560 037

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to Certify that the dissertation work **"ID Card Detection and Facial Recognition using OpenCV and Tensorflow"** carried out by Charan Kumar M J, Kushal Kumar B V, Kushal M, USN: 1CR16EC031, 1CR16EC067, 1CR16EC068, bonafide students of **CMRIT** in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the academic year **2019-20**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
___________________	___________________	___________________
Prof. Pappa M,	Dr. R. Elumalai	Dr. Sanjay Jain
Associate Professor,	Head of the Department,	Principal,
Dept. of ECE.,	Dept. of ECE.,	CMRIT,
CMRIT, Bengaluru.	CMRIT, Bengaluru.	Bengaluru.

External Viva

Name of Examiners	Signature & date
1.	
2	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose consistent guidance and encouragement crowned our efforts with success.

We consider it as our privilege to express the gratitude to all those who guided in the completion of the project.

We express my gratitude to Principal, **Dr. Sanjay Jain,** for having provided me the golden opportunity to undertake this project work in their esteemed organization.

We sincerely thank Dr. R. Elumalai**,** HOD**,** Department of Electronics and Communication Engineering, CMR Institute of Technology for the immense support given to me.

We express my gratitude to our project guide Prof Pappa M, Assistant Professor, for their support, guidance and suggestions throughout the project work.

Last but not the least, heartful thanks to our parents and friends for their support.

Above all, We thank the Lord Almighty for His grace on us to succeed in this endeavor.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

Object detection is a general term for computer vision techniques for locating and labeling objects. Object detection techniques can be applied both to static images or moving images. Computer vision techniques are already in wide use in every field today, as they can offer valuable insight about movement of individual object or whole items to training, help in tracking and help visualizing progression during object movement [1]. Open source libraries such as OpenCV's DNN library and TensorFlow Object Detection API offer easy-to-use, open source frameworks where pre-trained models for object detection (such as ones downloadable from the TensorFlow model zoo) reach high accuracy in detecting various object from humans to tv monitors [2].

Object detection is a term related to computer vision and image processing techniques for locating and annotating all the potential objects in the images. The images can be either static pictures or moving frames. The task of object detection is closely related to two other problems: image classification and object localization. Image classification also referred to as image recognition, is about classifying an image to a particular class out of all the possible classes. Object localization is more advanced than image recognition, and it demands to localize the labeled object in an image. Object detection is the most complex of all—it involves labeling and localizing multiple objects in an image. Object detection techniques are extensively used for applications related to face detection, locating pedestrian on streets or players on football grounds, and detecting vehicles on roads.

The computer vision community started growing enthusiasm towards deep learning models after the success of AlexNet, a deep convolutional neural network (CNN), at the ImageNet Challenge 2012 for image recognition [3]. The success of AlexNet led to the development of several other successful deep neural networks, including VGGNet [4], GoogleNet (Inception-v1) [5], and Residual Network (ResNet) [6] for the task of image classification. Since object detection is an extension of image recognition, most of the successful object detection models are based on deep neural networks for image classification. Consequently, most of the object detection models such as Faster Region-based Convolutional Neural Network (Faster R-CNN) [7], Region-based Fully Convolutional Network (R-FCN) [8], and Single-Shot Detector (SSD) [9] are built on top of image classification models such as ResNet and Inception. However, a major constraint of a deep CNN is that it requires a large amount of annotated datasets for training them. To overcome the problem of enormous training datasets, several public datasets such as the ImageNet database [10], the Common Objects in Context (COCO) dataset [11], and the Open Images dataset [12] are readily available. Moreover, the development of Graphical Processing Units (GPUs) helps in training the computationally expensive deep neural networks. Tensorflow Object Detection API,1 OpenCV's DNN library,2 and Microsoft Cognitive Toolkit3 provide open-source frameworks to construct, train and deploy object detection models.

We are using Tensorflow because of its popularity and ease of use. Tensorflow Object Detection API also presents pre-trained models for object detection. The pre-trained models are trained on some of the public datasets such as COCO, Kitti, Open Images, and Atomic Visual Actions (AVA) v2.1. The performance of the pre-trained models is sufficient for recognizing objects, such as a person, television and cars, but not good enough for detecting the food classes relevant for our purposes. Moreover, the pre-trained models do not even detect some of the food classes expected in our application. Therefore, it is essential to fine-tune the existing models for our particular datasets and application requirements.

Imagining a world with robots in our daily lives seemed unachievable a few years ago. Nowadays, with the swift technological advances, robots are getting familiarized with our daily activities and becoming a part of our society by holding positions, such as medical assistants, social workers, industrial workers, educational robots, pets, etc. A considerable progress is being made in this direction; many laboratories around the world carry out interdisciplinary research to develop robots that are useful and safe for everyday life. Moreover, they develop research around the interaction [1] between robots and people to guarantee efficient, ethical and responsible use. The face is our primary focus of attention in social life playing an important role in conveying identity and emotions. We can recognize a number of faces learned throughout our lifespan and identify faces at a glance even after years of separation. This skill is quite robust despite of large variations in visual stimulus due to changing condition, aging and distractions such as beard, glasses or changes in hairstyle. Computational models of face recognition are interesting because they can contribute not only to theoretical knowledge but also to practical applications. Computers that detect and recognize faces could be applied to a wide variety of tasks including criminal identification, security system, image and film processing, identity verification, tagging purposes andhuman-computer interaction. Unfortunately, developing a computational model of face detection and recognition is quite difficult because faces are complex, multidimensional and meaningful visual stimuli.

Face detection is used in many places now a days especially the websites hosting images like picassa, photobucket and facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person is in the image. In our project, we have studied and implemented a pretty simple but very effective face detection algorithm and recognition algorithm using LBPH which takes human skin colour into account. Our aim, which we believe we have reached, was to develop a method of face recognition that is fast, robust, reasonably simple and accurate with a relatively simple and easy to understand algorithms and techniques after custom ID card detection using tensorflow for attendance system. To test and refine the pre-trained models, we are using user defined database for ID card as well as facial image.Moreover, the user database has bounding box data for some of the datasets, and the bounding box data for each class dataset is labeled for that particular class only. However, object detection requires bounding box data to be annotated for all the possible classes and not just for one class. So, the I bounding box data requires pre-processing before it can be used for training and evaluating an object detection model.To improve the accuracy of object detector further, we can also leverage the available contextual information. For instance, we may utilize the context like the list of objects present.

Chapter 2

LITERATURE SURVEY

Image classification and object detection are terms related to computer vision and image processing techniques. Image classification is classifying an image into a particular class, while object detection is about recognizing, locating and labeling objects in all the possible classes [13]. Object detection has been used in many applications, including: Human-Computer Interaction (HCI), visual computing, robotics, security, transportation, and consumer electronics. Deep learning started dominating computer vision since 2012, when Krizhevsky et al. [14] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by releasing AlexNet.

2.1 Conceptual Level

2.1.1 Artificial Neural Network (ANN)

Artificial Neural Network, loosely inspired by the neural structure of our brain, consists of a network of connected nodes known as neurons. Neurons are the central processing unit of the neural networks, and they are connected just like neurons and synapses in a biological brain. The weighted neural connections assist in transmitting signals (features) from one neuron to another [15]. Artificial neural networks learn by examples rather than programming them with task-specific rules. For instance, in visual pattern recognition, they might learn with training examples of handwritten digits from 0 to 9 and use this knowledge to identify digits in other unseen samples [16].Artificial neural networks (ANN) mostly consist of three types of layers, an input layer, an output layer, and one or more hidden layers. The first hidden layer learns simple, and basic features and passes them to the second hidden layer. Based on the inputs from the previous layer, the second layer learns features which are more complex and at a more abstract level than the first one. Similarly, layers farther in the network can learn more complicated features and so on. In this way, a multi-layered network can be used effectively to solve sophisticated problems like image classification, object detection, and others.

2.1.2 Deep Learning

Deep Learning, a subfield of machine learning and artificial intelligence, is related to the training of computational models that are composed of multi-layered artificial neural networks. The multi-layered ANN is known as a deep neural network (DNN). Deep in deep neural networks corresponds to the depth of the network. Deep networks usually have more than two layers of hidden neurons between the input and output layers [16]. The deep neural networks have enhanced the state-of-the-art accuracy in image classification, object detection, speech recognition, language translation, and other areas considerably [17]. Deep learning methods are based on learning representations (features) from data, such as text, images, or videos, rather than implementing task-specific algorithms [16]. Learning can either be unsupervised or supervised. However, most of the practical systems deploy supervised learning to leverage the benefits of deep learning [14]. Supervised learning, in essence, means learning from labeled data. Andrew Ng [13] pointed out the performance of deep learning methods increases with an increase in the amount of training data as opposed to traditional learning methods that saturate in performance; this characteristic makes the deep learning methods scalable.

2.1.3 Convolutional Neural network (CNN)

Convolutional Neural Network (CNN) is the most common class of architectures used for deep learning. CNN works similarly as artificial neural network except it has a series of convolutional layers at the beginning. CNN is widely used for visual recognition tasks such as image classification, object detection, and speech recognition. CNN has become ubiquitous among researchers and computer vision community ever since AlexNet showcased outstanding performance at the ImageNet Challenge in 2012 [14].

2.2 Model Level

Object detection is a common term for computer vision techniques classifying and locating objects in an image. Modern object detection is largely based on use of convolutional neural networks [2]. Some of the most relevant system types today are

Faster R-CNN, R-FCN, Multibox Single Shot Detector (SSD) and YOLO (You Only Look Once) [2]. Original R-CNN method worked by running a neural net classifier on samples cropped from images using externally computed box proposals (samples cropped with externally computed box proposals; feature extraction done on all the cropped samples).

This approach was computationally expensive due to many crops; Fast RCNN reduced the computation by doing the feature extraction only once to the whole image and using cropping on the lower layer (feature extraction only once on the whole image; samples cropped with externally computed box proposals). Faster RCNN goes a step further and used the extracted features to create class-agnostic box proposals (feature extraction only once on the whole image; no externally computed box proposals). R-FCN is like Faster R-CNN but the feature cropping is done in a different layer for increased efficiency [2]. YOLO (You Only Look Once) [18] works on different principle than the aforementioned models: it runs a single convolutional network on the whole input image (once) to predict bounding boxes with confidence scores for each class simultaneously.

The advantage of the simplicity of the approach is that the YOLO model is fast (compared to Faster R-CNN and SSD) and it learns a general representation of the objects. This increases localization error rate (also, YOLO does poorly with images with new aspect ratios or small object flocked together) but reduces false positive rate. [18] Single Shot Multibox Detector (SSD) [19] differs from the R-CNN based approaches by not requiring a second stage per-proposal classification operation [2]. This makes it fast enough for real-time detection applications. However, this comes with a price of reduced precision [2]. Note that “SSD with MobileNet” refers to a model where model meta architecture is SSD and the feature extractor type is MobileNet.

Speed-accuracy tradeoff Many modern object detection applications require real-time speed. Methods such as YOLO or SSD work that fast, but this tends to come with a decrease in accuracy of predictions, whereas models such as Faster R-CNN achieve high accuracy but are more expensive to run. The cost in model speed depends on the application: With larger images (e.g. 600x600) SSD works comparable to more computationally expensive models such Faster R-CNN, even as on smaller images its performance is considerably lower [2].

2.3 Proposed System

One of the main objectives of our research is to train custom object detectors and infer which one works best for detecting the three classes of our interest. For this purpose, it is important to select appropriate models of object detection and also a dataset that can be used for training and evaluation purposes. This chapter starts with a description of the dataset used throughout our research along with the pre-processing required on that dataset. Next, we explain the fine-tuning of the three Tensorflow pre-trained models of object detection.

2.3.1 Data collection and Preprocessing

Typically one of the significant challenging aspects of a deep CNN is to collect labeled examples to train image classifiers. In general, supervised learning demands thousands of annotated training examples. To meet the requirement of large training data and benchmark evaluation, the number of publicly available datasets are proliferating. In the context of computer vision, the available datasets contain large sets of images with additional information such as annotations, segmentation masks or other contextual data. Since there are several such datasets available for training and validation, it is essential to select the one which serves our purposes best. We selected the user defined database because it contains our own images for each class along with bounding box data for some images. Moreover, the user database has been used successfully for various applications of image classification which is an integral. To train and evaluate our object detection models, we first extracted the image and bounding box datasets for three classes namely, id card , tag and phone. Some of the issues we encountered with the extracted datasets are as follows:

- The user defined datasets consist of images of all sizes. However, we want to make sure that the images used for training and evaluation are not too small (i.e., at least 300 pixels).
- Another problem with the extracted datasets is that the bounding box data of each class is labeled for that particular class only and not all possible classes.

- Additionally, the number of bounding box examples expected for each class is insufficient. For the training, validation, and evaluation purposes, we require at least 620 bounding box examples for each class.

2.3.2 Selection and Annotation

We selected and annotated around 100 examples for each class. The selection process is fairly simple but, the annotation part of manual labeling of images required a significant amount of time. To annotate the images, we used a tool called LabelImg, which is a Python-based annotation tool for labeling graphical images. The annotations in the form of bounding box coordinates, and labels are saved as XML (Extensible Markup Language) files in PASCAL VOC (Visual Object Classes) or YOLO (You Only Look Once) format. The bounding box datasets of the ImageNet database are also in the form of XML files in PASCAL VOC format. Hence, we used the PASCAL VOC format to save the annotations using this tool.

2.3.3 Dataset Creation

After completing the annotation process, we divided each class dataset into three sets: training dataset, and test dataset. Each training dataset contains about 270 examples, and test dataset have around 60 samples. Then, the corresponding datasets for all the classes are merged—three training datasets for three classes. We then combined the three training datasets into one training dataset with the training examples from all the classes. Similarly, we combined the validation and test datasets as well.

2.3.4 TF Record Creation

To fine-tune a pre-trained Tensorflow object detection model, the training and validation datasets are converted into a TFRecord format. TFRecord is a standard format, supported by Tensorflow for training object detection models. Initially, the training and validation datasets are in XML files format; they are first converted into CSV (Comma Separated Values) file format, and then into a TFRecord format.Tensorflow models are pre-trained for multiple datasets such as COCO, Kitti, and Open Images. We selected three pre-trained models trained on the COCO dataset to fine-tune them with the images from the ImageNet database.

2.3.5 API Fine Tuning

As stated earlier, the pre-trained models for object detection are not good at detecting all the three classes required for our research. So, there is a need to fine-tune the pretrained models with our dataset to be able to detect those three classes effectively. The reason for fine-tuning a pre-trained model is that if a model is trained from scratch with randomly initialized weights, it might take weeks or months for the model to converge. Even then, the accuracy might not be good. This process of fine-tuning a pre-trained model is known as transfer learning, and it takes comparatively littletime to train a model with transfer learning. We selected one pre-trained model for object detection namely, SSD with MobileNetv2.

2.3.6 ID Card Detection with SSD Mobile Net

The pre-trained SSD-MobileNet-v2 model (ssd mobilenet v2 coco) is fine-tuned with the training and validation datasets created earlierusing the provided configuration file (ssd mobilenet v2 coco.config) as the basis of our fine-tuning. The configuration file requires a checkpoint file to initiate the finetuning process which is already provided by Tensorflow for SSD-MobileNet-v2.4 The configuration file also requires training record and test record file locations. Training record is a TFRecord file created for training dataset, and a test record is a file created for validation dataset. Another requisite of the configuration file is the location of an object label map file. The structure of an object label map file is similar, and ends with '.pbtxt' extension. The training record and test record file locations are added along with the object label map file. Other important parameters mentioned in the configuration file include:

• number of layers: 6

• aspect ratios: 0.33, 0.5, 1.0, 2.0, and 3.0

• learning rate: initial value is 0.004 with a decay factor of 0.95 after 800,720

steps (iterations) of training

• data augmentation options: random horizontal flip and ssd random crop

• batch size: 24

The SSD model is fine-tuned for 80,000 steps. Classification loss is based on how good an object is classified; localization loss is based on how well a correctly classified object is localized. The total loss values are added to the training log files after a particular interval of time. The log files are then passed as input to Tensorboard, a visualization tool to plot the computational graphs.

2.3.7 Evaluation Metrics

The evaluation metrics used to evaluate performance in this thesis are Precision, Recall, Precision-Recall curves (PR curves), and Mean Average Precision (mAP). Precision, Recall, PR curves are relevant measures for binary classification. However, in this thesis, we are concerned with eight classes. So, precision, recall, and PR curves are calculated for each class, separately as opposed to all the other classes.

2.3.8 Face Recognition

The face detection algorithm introduced in this section is an extension of Viola and Jones system based on boosted cascades of Haar-like features. As pointed out by Zhang et al. [19], these features are very efficient early in the cascade to quickly discard most of the background regions. However, in the last stages of the cascade, a large number of Haar-like features (several hundreds) are necessary to reach the desired detection/false acceptance rate trade-off. It results in a long training procedure and cascades with several dozens of stages which are difficult to design. Furthermore, Haar-like features are not robust to local illumination changes. To cope with the limitation of Haar-like features, we propose to use LBP features. The method to build the weak classifiers is inspired by the work of Fröba and Ernst and the cascade training is done with AdaBoost. For face recognition, data collection, training and testing are done seperately.

Chapter 3

NEURAL NETWORKS

Object detection using complex machine learning techniques is possible due to the recent advances in image classification using convolutional neural networks and improvements in hardware. This chapter aims to give the reader a basic understanding of how a convolutional neural network works and how it classifies images, explaining some key concepts, without going into too much detail. Artificial Neural Networks A neural network is a group of neurons or nodes connected to each other. Each node can receive a signal from its input nodes, process it, and pass it on to its output nodes.In an Artificial Neural Network (ANN), the nodes are structured in layers, where there is one input layer, one output layer, and one or more hidden layers in between. When a node receives a signal, it processes it using an activation function, which, given the input, defines the output of the node. The goal is to get the nodes to work together such that the network as a whole gives the correct output.

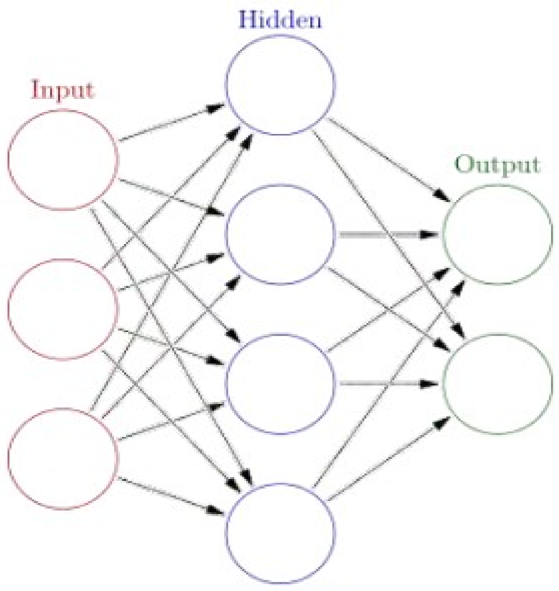


Figure.1 Artificial Neural Network

3.1 Training of Neural Networks

The process of training neural networks is to tune the parameters of the activation function of each individual node such that the final output of the network is correct. A simple example that would conform with figure 1, is input nodes where we have different features of a fish, e.g. its weight, color and age. The input nodes passes on the values of the fish features to the hidden layer, which processes the values using the activation function and passes on the value to the output nodes. The output nodes would be the fish type, e.g. salmon and cod. Ideally, if the input is in fact a salmon, we want the value of the salmon output node to get the value 1 and the cod output node to get the value 0. Deviations from this result can be computed as a loss. We want to find the parameters for the activation functions in all the nodes that minimizes this loss. This is achieved by training the nodes on data where the class of the input object is known. If we have a data set of 1,000 cods and 1,000 salmons and their weight, color, and age we can use these as inputs, pass it through the network, and look at the loss. Then, by using gradient descent, we can change the parameters in all the activation functions in a way that decreases the loss. This is done using backpropagation, which is an algorithm that calculates how the parameters in each activation function should be changed in order to reduce the loss. The idea is that doing this many times makes the network able to classify fish correctly, also on new data.

3.2 Convolutional Neural Networks

A convolutional neural networks is a type of artificial neural networks, and consist of nodes with activation functions structured into layers. The main difference is that it is assumed that the input is an image, and use this assumption to do operations that are tailored for images, and improve both runtime and accuracy for image classification. If we want to process an image through an artificial neural network, we would transform the image into a vector, where each row of the image now is in one large row. That would mean that color images, which has three channels (RGB) of size 256 x 256 would have 256 x 256 x 3 = 196,608 neurons in the first hidden layer. Convolutional neural networks reduces the number of neurons by using convolutional layers and pooling layers.

3.2.1 Convolutional Layers

A convolutional layer consists of a set of filters. When a convolutional layer receives an image input, it slides, or convolves, filters over all the pixels in the input image and output the dot product of the filter and the image at the filters position. This will create an activation map, where we can see the response of the filter at different positions. During training the different filters will change in order to detect different features in the input image. Imagine we want to classify images of handwritten ones and zeros. Some filters may aim to detect the rounded shape of the zeros, and will therefore output a large number if it detects a rounded shape. The dot product, or the activation of the two filters at this position in the image is $(30\cdot1)+ (10\cdot1)+(10\cdot1)+(10\cdot1) = 60$ for filter 1, and $(30\cdot1)+(30\cdot1)+(30\cdot1)+(30\cdot1)+(30\cdot1) = 150$ for filter 2. This means that the activation map for filter 2 is much higher than for filter 1 at this position. For a different subregion the result would be different. When we have several different filters they will activate at different positions and thus have different activation maps. Combining the different activation maps for the different filters tells us something about the image as a whole. During training of a convolutional neural network, the network is trained to draw conclusions of what is in the image based on which filters activate at which positions.

Pooling Layers

The pooling layers are downsampling layers. They work by sliding boxes of e.g. 2x2 over the pixels of the image, and process the values to output a single value for those 4 pixels. There are different ways to process the pixels, where max pooling is a commonly used method. An example of how to downsample an image by a factor of two using max pooling is shown in figure 2. Max pooling transforms the pixels inside the sliding box into one pixel with the same value as the highest pixel value in the original box. By downsampling the image the number of parameters is reduced, which in turn reduces the computational cost of the algorithm. The pooling layers usually follow a convolutional layer, and can therefore help the next convolutional layer pick up features the previous layer could not. Pooling layers can also help preventing overfitting, which is when a classification algorithm performs well on training data, but badly on test data.

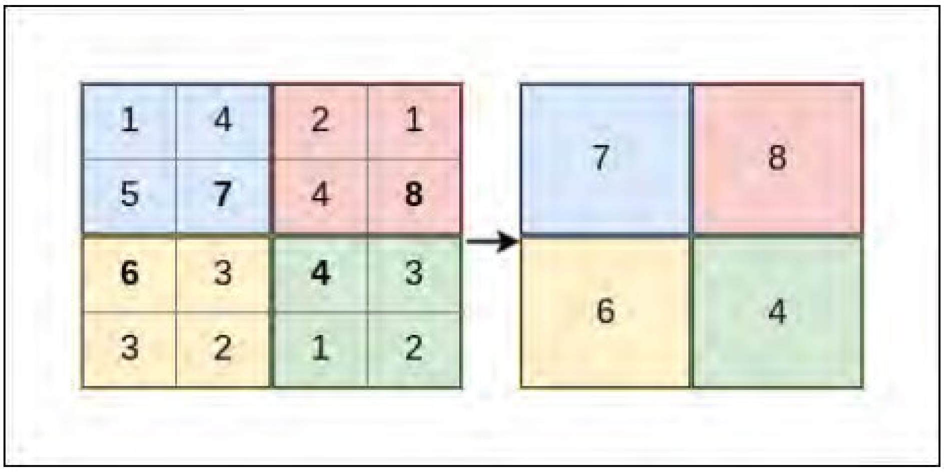


Figure.2 Pooling Layers

3.3 Transfer Learning

Transfer learning is a machine learning method where a network trained on one task is repurposed on a second similar task. This is a technique used extensively in this project. A problem in deep learning is data dependence, meaning that deep learning processes depends on a massive training dataset to be able to understand the patterns in the data. This is especially a problem in deep learning compared to traditional machine learning methods, since deep learning methods have to learn features from the dataset, while in traditional machine learning methods the user has to design the features. To counter the data dependence of a neural network one needs a large well-annotated dataset, which may be both costly and time consuming to acquire. Transfer learning provides a way of utilizing training data that does not conform with the test data. This means that e.g. a classifier trained to detect dogs can be retrained to detect ducks, saving training time and training data requirements compared to training a duck classifier from scratch.

Instances-based

Supplement the training data in the target domain with relevant instances from the source domain by assigning fitting weights to the selected values.

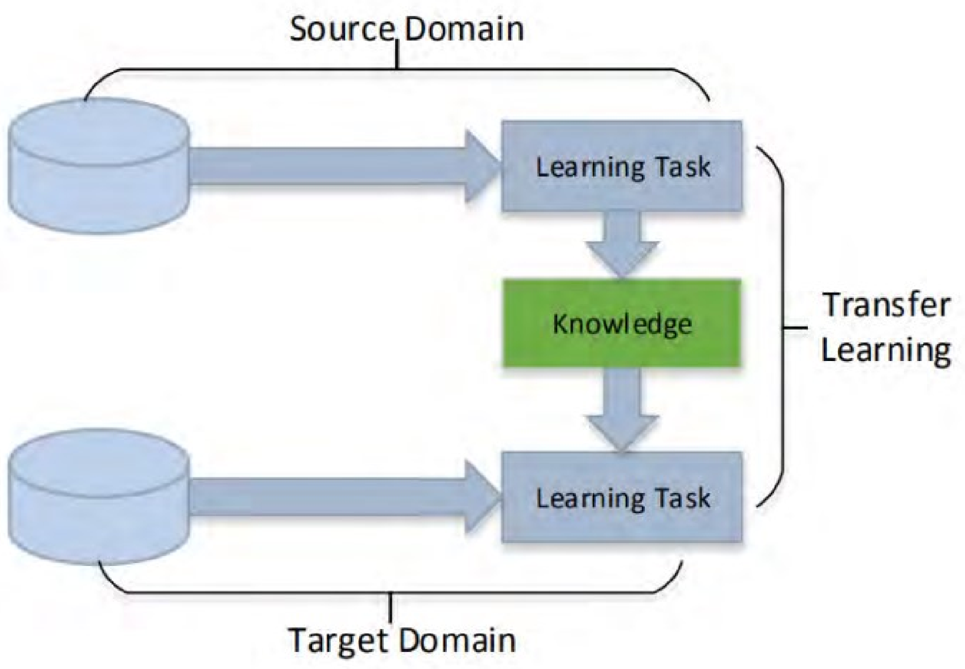


Figure.3 Transfer Learning

Network-based

Reuses parts of the network that are pre-trained on the source domain. The target domain retrains the last part of the network on new data while keeping the weights in the other layers. The idea is that the first layers in neural network find more general features, while the last layers narrow down to the features specific for the class of interest. A classifier trained on images of boats and a classifier trained on cars may detect similar features in the first layers of the network. This is project uses pre-trained weights to further train a boat detector.

Adversial-based

Finds transferable features that are suitable in both target domain and source domain.

SSD - Single shot Multibox Detector:

The Single Shot Multibox Detector works similarly as YOLO compared to Faster R-CNN. SSD does not use a region proposal network, but as YOLO, it discards the proposal generation step and does all the computation in a single network. A difference between YOLO and SSD is that SSD uses different aspect ratios and scales when it grids the input image, which makes it easier to handle objects of different sizes. By circumventing the proposal generation, SSD saves computational time and runs at a frame rate six to eight times higher than Faster R-CNN with similar accuracy, according to Liu et al. Figure 3 shows how SSD detects objects with different sizes using different aspect ratios on the feature map. Each cell in the feature maps tries different anchors, that is proportional to the cell size. The anchors are the default boxes shown with dotted lines in figure 3 The smallest ground truth object in the image was found in the eight by eight grid, while the bigger object is found using the four by four grid, illustrating how different grid sizes detects objects of different size. The anchors in each cell are run through a classifier which gives a confidence score. If multiple boxes predict the same class and have an intersect over union of more than 0.5 the bounding box that has the highest confidence score is chosen. This process is called non-maximum suppression. All the region-based object detection algorithms discussed work in two stages: first, generate region proposals and then classify those regions in a separate step. SingleShot Detector, an object detection algorithm proposed by Liu et al. [20] in 2016 predicts the bounding box (region) and the class simultaneously in a single shot. Doing both the operations in one step makes SSD faster and a viable option for real-time detections.

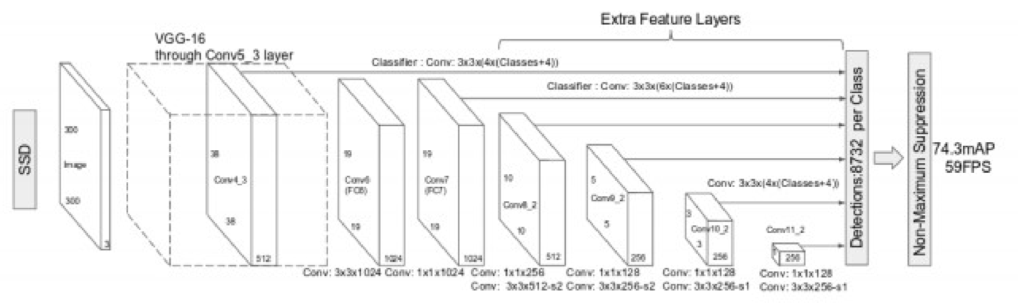


Figure.4 SSD Architecture

The architecture of SSD is shown in Figure. The working is described below:

- For the base network, the author has chosen the truncated version of VGG16, which is an image classification architecture. VGG16 is truncated to remove the classification part and extended with extra feature layers with sizes in decreasing order. An input image passes through all the convolution layers to generate multiple convolutional feature maps of different sizes.

- Then, for each of the feature maps produced above, anchor boxes are generated in a similar way as for Faster R-CNN. A convolution filter of size say, 3x3 is slid over each feature map to create anchor boxes at different scales and aspect ratios. Each predicted box has four coordinates and a vector of class probabilities.
- At the training time, all the predicted boxes are matched with the ground truth boxes based on IoU (Intersection over Union). As a result, all the boxes with IoU > 0.5 are considered as positives, and others as negatives.
- Since there are multiple feature maps of different sizes, the number of bounding boxes generated are quite large. Also, most of the anchor boxes are negatives.
- To overcome the shortcomings mentioned above, firstly non-maximum suppression is deployed at the end to combine the overlapping boxes. Secondly, the technique of Hard Negative Mining (HNM) is used to balance the negative and positive training examples. In training, only a subpart of negative examples is used. The subpart is created by sorting all the negatives by the confidence loss and selecting the top ones such that the ratio of negatives to positives is 3:1.

To conclude, SSD predicts the bounding boxes and class probabilities in one shot, which makes it faster than the region-based algorithms discussed above. Besides, accuracy is comparable to Faster R-CNN. Moreover, the variable sizes of the feature maps assist in generating boxes of varying dimensions, which in turn helps detect objects of different sizes [20]

3.4 Tensorflow Object Detection API

Google's Tensorflow Object Detection API is an open source framework for object detection; it is based on Tensorflow and allows a user to define, train and utilize the models for object detection. The Tensorflow open source software library, developed by Google Brain Team, utilizes data flow graphs for numerical computations [21]. Data flow graphs consist of two important components, nodes, and edges. Nodes represent the mathematical computations (operations), and edges represent tensors (multi-dimensional arrays) that flow between the nodes. Tensorflow is widely used to develop applications with deep learning [22, 23]. Another interesting feature of Tensorflow is that it supports a tool known as Tensorboard for in-depth visualization of models during the training process. Tensorboard provides a web interface for visualization of computational graphs and to understand how the parameters and performance change while training a model. As mentioned earlier, Tensorflow Object Detection API provides multiple pretrained models such as SSD model with MobileNet, Faster R-CNN model with ResNet, and R-FCN model with ResNet on different datasets. To fine-tune our specific object detectors, we can choose among the pre-trained models to initialize our training. The selection of pre-trained model is based on the purpose of our application. The API also gives an insight into speed and accuracy trade-offs of different object detection models 9. The most important requisite of modern object detection applications is to achieve real-time speed. The models trained with SSD and R-FCN networks are pretty fast, but this speed comes at the cost of reduced accuracy. On the contrary, models trained with Faster R-CNN are more accurate but are expensive in time [24]. However, the models trained with SSD and R-FCN are faster but less accurate. Moreover, Faster R-CNN can be fast enough, if the regions of interest are limited as evident for Faster R-CNN with ResNet with 50 proposals.

3.5 Face Recognition

Correlation methods require substantial computation time and enormous amounts of storage. Therefore, features reduction and face representation are needed in the face recognition system. LBPH is usually a preferred method in computer vision, image processing, and pattern recognition; it is appropriate for the feature because it describes the texture and structure of an image. We represent the face image and reduce the image dimension by applying the LBPH method, extracting the features texture of the image by dividing the image into local regions and extracting the binary pattern for each local region. The original LBP operator, which works on eight neighbors of a pixel, was introduced by Ojala et al. [27]. The image is divided into small regions called the cell. Each pixel in the cell is compared with each of its eight neighbors. The center pixel value will be used as the threshold value. The eight-neighbors-pixel will be set to one if its value is equal to or greater than the center pixel; otherwise, the value is set to zero. Accordingly, the LBP code for the center pixel is generated by concatenating the eight neighbor pixel values (ones or zeroes) into a binary code, which is converted to a 256-

dimensional decimal for convenience as a texture descriptor of the center pixel. We used a modified LBP operator called a uniform pattern. The pattern is the number of bitwise transitions from 1 to 0 or vice versa. The LBP is called uniform if its uniformity measure is at most 2. We proposed in this paper an enhanced Face Recognition Framework Based on Correlated Images.

Chapter 4

HARDWARE

4.1 RASPBERRY PI 3 MODEL B+

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. Several generations of Raspberry Pis have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM compatible central processing unit (CPU) and on-chip graphics processing unit (GPU).

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+; on-board memory ranges from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either SDHC or MicroSDHC sizes. The boards have one to four USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm phono jack for audio output. Lower-level output is provided by a number of GPIO pins which support common protocols like I^2C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.

The first generation (Raspberry Pi 1 Model B) was released in February 2012, followed by the simpler and cheaper Model A. In 2014, the Foundation released a board with an improved design, Raspberry Pi 1 Model B+. These boards are approximately credit-card sized and represent the standard mainline form-factor. Raspberry Pi 3 Model B was released in February 2016 with a 64 bit quad core processor, and has on-board WiFi, Bluetooth and USB boot capabilities.

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Everyone knows and loves the Raspberry Pi, but what if the wireless capabilities only got better? The Raspberry Pi 3 B+ is here to provide you with the same Pi as before, but now with gigabit and PoE capable Ethernet, as well as better overheating protection for the 64-bit processor. The credit-card-sized computer is capable of many of the things your desktop PC does, like spreadsheets, word processing and playing high-definition video and games. It can run several flavors of Linux (and even Windows 10 free-of-charge) and is being used to teach kids all over the world how to program.

The secret sauce that makes this computer so small and powerful is the Broadcom BCM2837, an ARM Cortex-A53 64-bit Quad Core Processor System-on-Chip operating at 1.4GHz. The GPU provides OpenGL ES 2.0, hardware-accelerated OpenVG and 1080p30 H.264 high-profile decode. It is capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs of general-purpose compute. What does that all mean? It means that if you plug the Raspberry Pi 3 B+ into your HDTV, you could watch Blu-ray quality video, using H.264 at 40MBits/s.

The Raspberry Pi 3 B+ has four built-in USB ports that provide enough connectivity for a mouse, keyboard or anything else that you feel the RPi needs. But if you want to add even more, you can still use a USB hub. Keep in mind, it is recommended that you use a powered hub so as not to overtax the onboard voltage regulator. Powering the Raspberry Pi 3 B+ is easy: just plug any 5V/2.5A USB power supply into the microUSB port. There's no power button, so the RPi will begin to boot as soon as power is applied. To turn it off, simply shut down the Pi 3 B+, then remove power. The four built-in USB ports can even output up to 1.2A, enabling you to connect more power-hungry USB devices.

On top of all that, the low-level peripherals on the RPi make it great for hardware hacking. The 0.1" spaced 40-pin GPIO header on the RPi gives you access to 27 GPIO, UART, I2C, SPI, as well as 3.3 and 5V sources. Each pin on the GPIO header is identical to its predecessor, the Model 3. If you are planning to run a NOOBS card with the RPi3 B+ make sure that it is up to date with the latest version.

4.2 Specification

1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT)

- Broadcom BCM2837B0 64-bit ARM Cortex-A53 Quad Core Processor SoC running @ 1.4GHz
- 1GB RAM LPDDR2 SDRAM
- 4x USB2.0 Ports with up to 1.2A output
- Extended 40-pin GPIO Header
- Video/Audio Out via 4-pole 3.5mm connector, HDMI, CSI camera, or Raw LCD (DSI)
- Storage: microSD
- Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
- Low-Level Peripherals:
  27x GPIO
  UART
  I2C bus
  SPI bus with two chip selects
  +3.3V
  +5V
  Ground
- Power Requirements: 5V @ 2.5A via microUSB power source
- Supports Raspbian, Windows 10 IoT Core, OpenELEC, OSMC, Pidora, Arch Linux, RISC OS and More!
- 85mm x 56mm x 17mm

4.3 Technical Specification

Processor

- Broadcom BCM2837B0 64-bit ARM Cortex-A53 Quad Core Processor SoC running @ 1.4GHz

- 1GB RAM LPDDR2 SDRAM
- 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
- IEEE 802.11 b / g / n Wi-Fi. Protocol: WEP, WPA WPA2, algorithms AES-CCMP (maximum key length of 256 bits), the maximum range of 100 meters.
- IEEE 802.15 Bluetooth, symmetric encryption algorithm Advanced Encryption Standard (AES) with 128-bit key, the maximum range of 50 meters.

GPU

- Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated
- Open VG, and 1080p30 H.264 high-profile decode.
- Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure

Memory

- 1GB LPDDR2

Operating System

- Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT

Dimensions

- 85 x 56 x 17mm

Power

- Micro USB socket 5V1, 2.5A

Connectors:

- Ethernet: 10/100 BaseT Ethernet socket
- Video Output: Composite RCA (PAL and NTSC)
- Audio Output 3.5mm jack
- HDMI (rev 1.3 & 1.4)
- USB 4 x USB 2.0 Connector
- GPIO Connector: 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
- Camera Connector: 15-pin MIPI Camera Serial Interface (CSI-2)
- Display Connector: Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
- Memory Card Slot: Push/pull Micro SDIO

The GPU provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode and is capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs of general purpose compute. What's that all mean? It means that if you plug the Raspberry Pi 3 into your HDTV, you could watch BluRay quality video, using H.264 at 40MBits/s.

The biggest change that has been enacted with the Raspberry Pi 3 is an upgrade to a next generation main processor and improved connectivity with Bluetooth Low Energy (BLE) and BCM43143 Wi-Fi on board. Additionally, the Raspberry Pi 3 has improved power management, with an upgraded switched power source up to 2.5 Amps, to support more powerful external USB devices.

The Raspberry Pi 3's four built-in USB ports provide enough connectivity for a mouse, keyboard, or anything else that you feel the RPi needs, but if you want to add even more you can still use a USB hub. Keep in mind, it is recommended that you use a powered hub so as not to overtax the on-board voltage regulator. Powering the Raspberry Pi 3 is easy, just plug any USB power supply into the micro-USB port. There's no power button so the Pi will begin to boot as soon as power is applied, to turn it off simply remove power. The four built-in USB ports can even output up to 1.2A enabling you to connect more power hungry USB devices (This does require a 2Amp micro USB Power Supply).

On top of all that, the low-level peripherals on the Pi make it great for hardware hacking. The 0.1" spaced 40-pin GPIO header on the Pi gives you access to 27 GPIO, UART, I 2C, SPI as well as 3.3 and 5V sources. Each pin on the GPIO header is identical to its predecessor the Model B+.

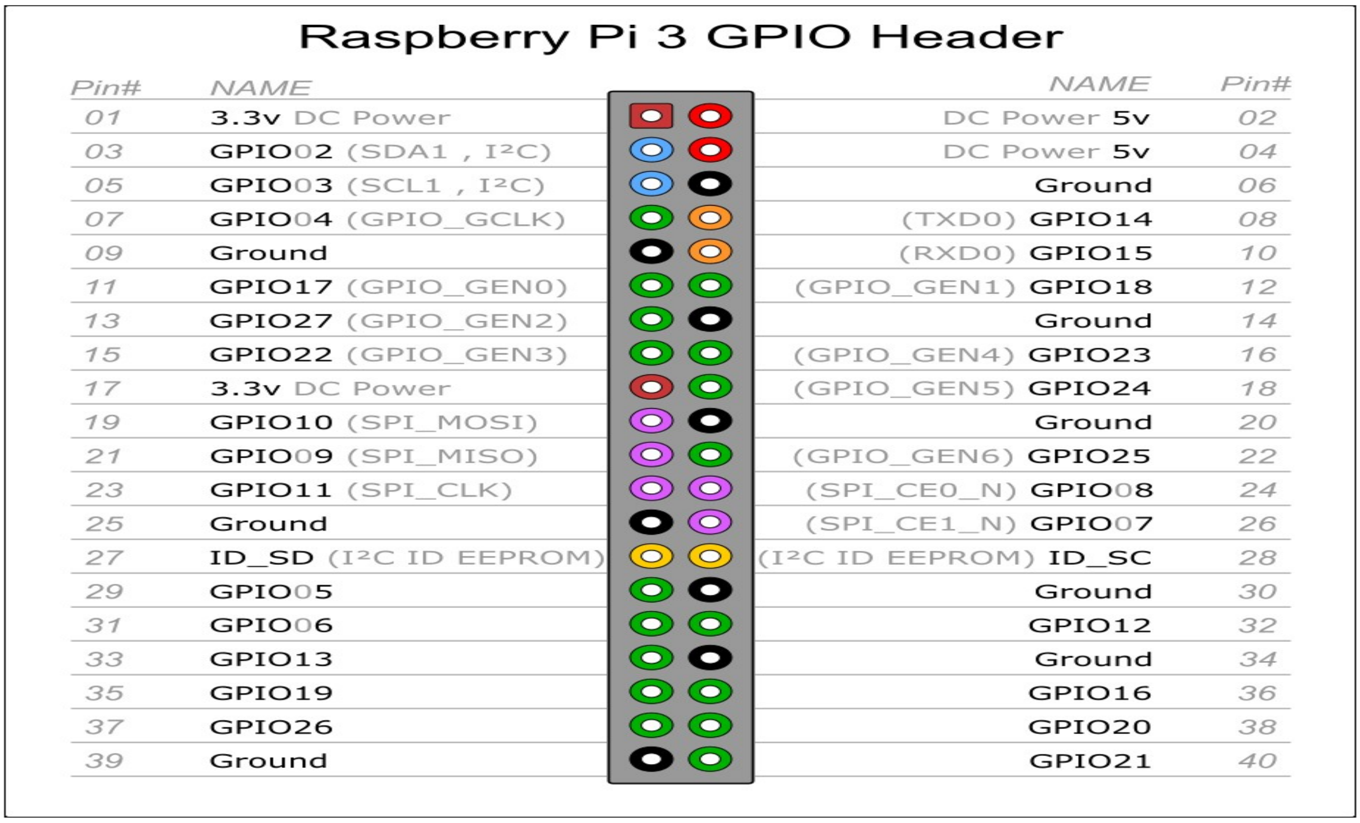
SoC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a VideoCore IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.

Figure.5 SoC

GPIO

The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pis going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO's pins, but that's handled internally by the operating system.



Raspberry Pi 3 GPIO Header

Pin#	NAME			NAME	Pin#
01	3.3v DC Power	🟥	🔴	DC Power 5v	02
03	GPIO02 (SDA1 , I^2C)	🔵	🔴	DC Power 5v	04
05	GPIO03 (SCL1 , I^2C)	🔵	⚫	Ground	06
07	GPIO04 (GPIO_GCLK)	🟢	🟠	(TXD0) GPIO14	08
09	Ground	⚫	🟠	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	🟢	🟢	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	🟢	⚫	Ground	14
15	GPIO22 (GPIO_GEN3)	🟢	🟢	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	🔴	🟢	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	🟣	⚫	Ground	20
21	GPIO09 (SPI_MISO)	🟣	🟢	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	🟣	🟣	(SPI_CE0_N) GPIO08	24
25	Ground	⚫	🟣	(SPI_CE1_N) GPIO07	26
27	ID_SD (I^2C ID EEPROM)	🟡	🟡	(I^2C ID EEPROM) ID_SC	28
29	GPIO05	🟢	⚫	Ground	30
31	GPIO06	🟢	🟢	GPIO12	32
33	GPIO13	🟢	⚫	Ground	34
35	GPIO19	🟢	🟢	GPIO16	36
37	GPIO26	🟢	🟢	GPIO20	38
39	Ground	⚫	🟢	GPIO21	40

Figure.6 Raspberry Pi 3 GPIO header

USB chip

The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to-Ethernet adaptor and USB hub.

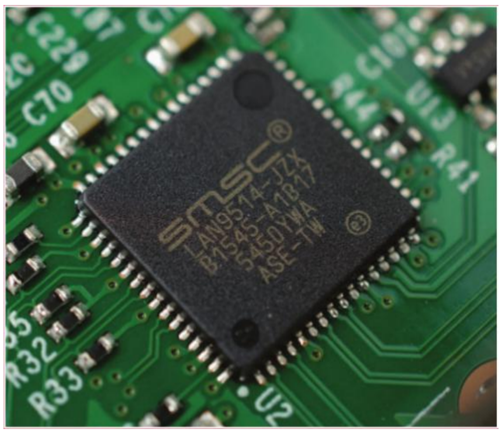


Figure.7 USB chip

Antenna

There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.

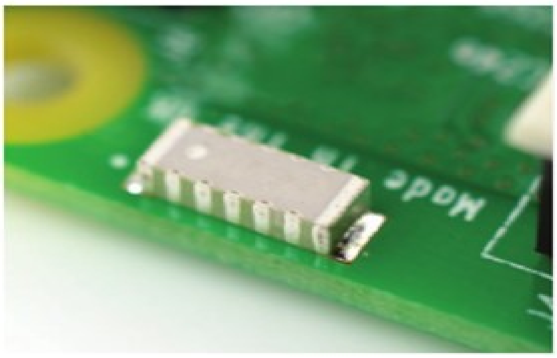


Figure.8 Antenna

Key Improvements from Pi 2 Model B to Pi 3 Model B:

• Next Generation QUAD Core Broadcom BCM2837 64bit ARMv7 processor

• Processor speed has increased from 900MHz on Pi 2 to 1.25Ghz on the RPi 3 Model B

• BCM43143 Wi-Fi on board

• Bluetooth Low Energy (BLE) on board

• Upgraded switched power source up to 2.5 Amps (can now power even more powerful devices over USB ports)

The main differences are the quad core 64-bit CPU and on-board Wi-Fi and Bluetooth. The RAM remains 1GB and there is no change to the USB or Ethernet ports. However, the upgraded power management should mean the Pi 3 can make use of more power hungry USB devices

For Raspberry Pi 3, Broadcom have supported us with a new SoC, BCM2837. This retains the same basic architecture as its predecessors BCM2835 and BCM2836, so all those projects and tutorials which rely on the precise details of the Raspberry Pi hardware will continue to work. The 900MHz 32-bit quad-core ARM CortexA7

CPU complex has been replaced by a custom-hardened 1.2GHz 64-bit quad-core ARM Cortex-A53

In terms of size it is identical to the B+ and Pi 2. All the connectors and mounting holes are in the same place so all existing add-ons, HATs and cases should fit just fine although the power and activity LEDs have moved to make room for the WiFi antenna.

The performance of the Pi 3 is roughly 50-60% faster than the Pi 2 which means it is ten times faster than the original Pi.

All of the connectors are in the same place and have the same functionality, and the board can still be run from a 5V micro-USB power adapter. This time round, we’re recommending a 2.5A adapter if you want to connect power-hungry USB devices to the Raspberry Pi.

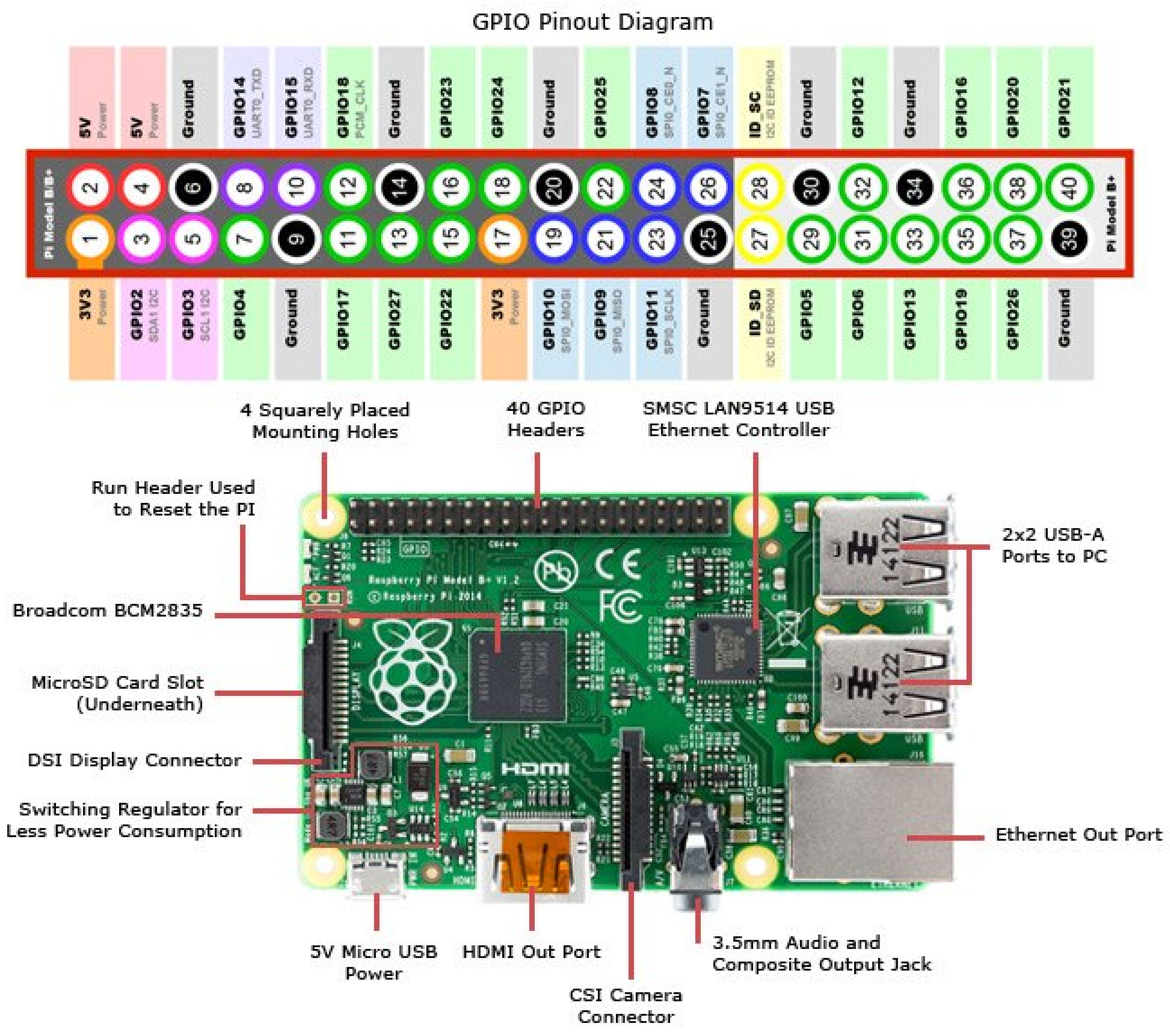


Figure.9 GPIO pinout diagram

Chapter 5

SOFTWARE

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until July 2018.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python features a comprehensive standard library, and is referred to as "batteries included".

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model. Python and CPython are managed by the non-profit Python Software Foundation.

5.1 Features

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

5.2 Applications

- web development (server-side),
- software development,
- mathematics,
- system scripting.

5.3 What can Python do

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

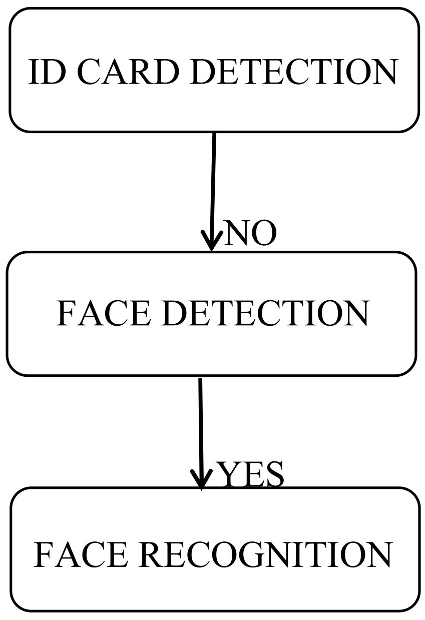
5.4 Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-orientated way or a functional way.

Chapter 6

WORKFLOW



6.1 ID card detection

In this process we will identify if a person is or not wearing an ID card. This is done using a process called object detection using tensorflow.Google's Tensorflow Object Detection API is an open source framework for object detection; it is based on Tensorflow and allows a user to define, train and utilize the models for object detection. The Tensorflow open source software library, developed by Google Brain Team, utilizes data flow graphs for numerical computations [21].

This process is achieved using the following steps:

- Data collection
- Image labeling
- Training data generation
- Label mapping
- Training configuration

- Inference graph
- ID card detection

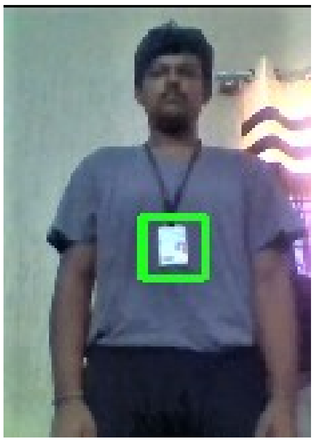


Figure.10 ID card detection

6.2 Face detection

This process is to find all the possible faces in the video frame. This process is done only if the person is not wearing the ID card. We use OpenCV harrcascade method to perform this. The concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

This process involves the following steps:

- Data collection
- Collecting positive images (images containing faces)
- Collecting negative images (images without faces)
- Training the system with positive and negative images
- Generating an XML file
- Face detection

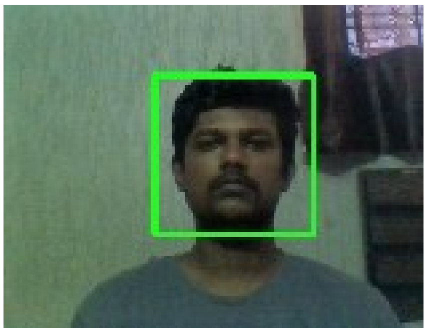


Figure.11 Face detection

6.3 Face recognition

This process is to identify whose face it is in the detected face in previous section.Correlation methods require substantial computation time and enormous amounts of storage. Therefore, features reduction and face representation are needed in the face recognition system. LBPH is usually a preferred method in computer vision, image processing, and pattern recognition; it is appropriate for the feature because it describes the texture and structure of an image.

The process is performed using following steps:

- Collecting face data
- Labeling the face data
- Generating a YML file
- Face detection
- Face recognition

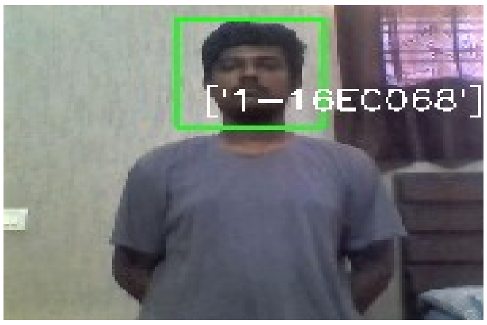


Figure.12 Face recognition

Chapter 7

RESULTS

There are two cases in this system case 1 is when the person is wearing ID card and case 2 is when the person is not wearing ID card. The results for both the cases are derived below

7.1 With ID card

Case 1: Person wearing ID card

The main function of the project is to find if the person is wearing ID card or not and this is done by object detection using tensorflow. Now if the person is wearing ID card the system should not do anything, it will just mark the ID card with a green rectangle as shown in the figure below.

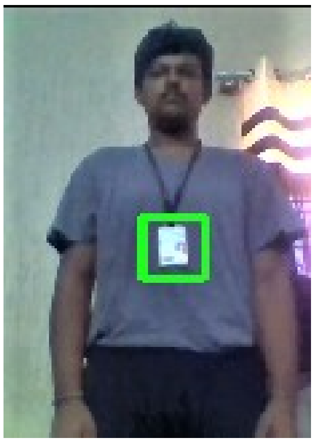


Figure.13 Person wearing ID card

7.2 Without ID card

Case 2: Person not wearing ID card.

The following steps are followed.

Step 1: ID card not detected.

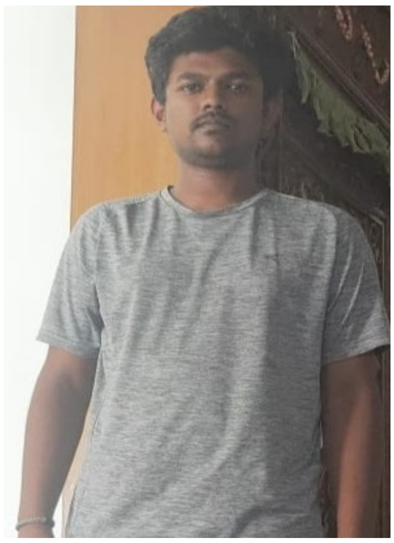


Figure.14 Person not wearing ID card

Step 2: Detect face in the image.

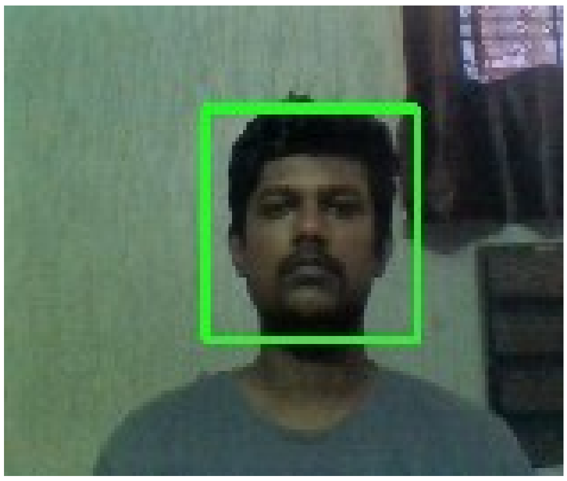


Figure.15 Face detected

Step 3: Identify the face in the image.

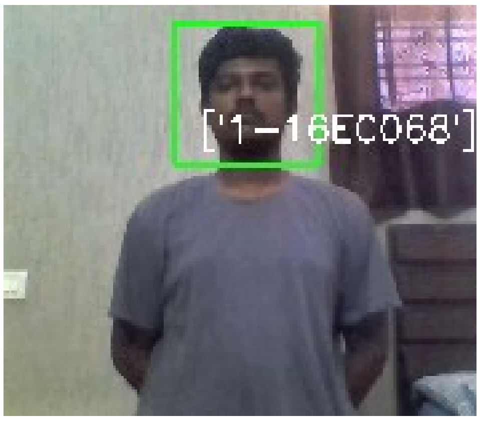
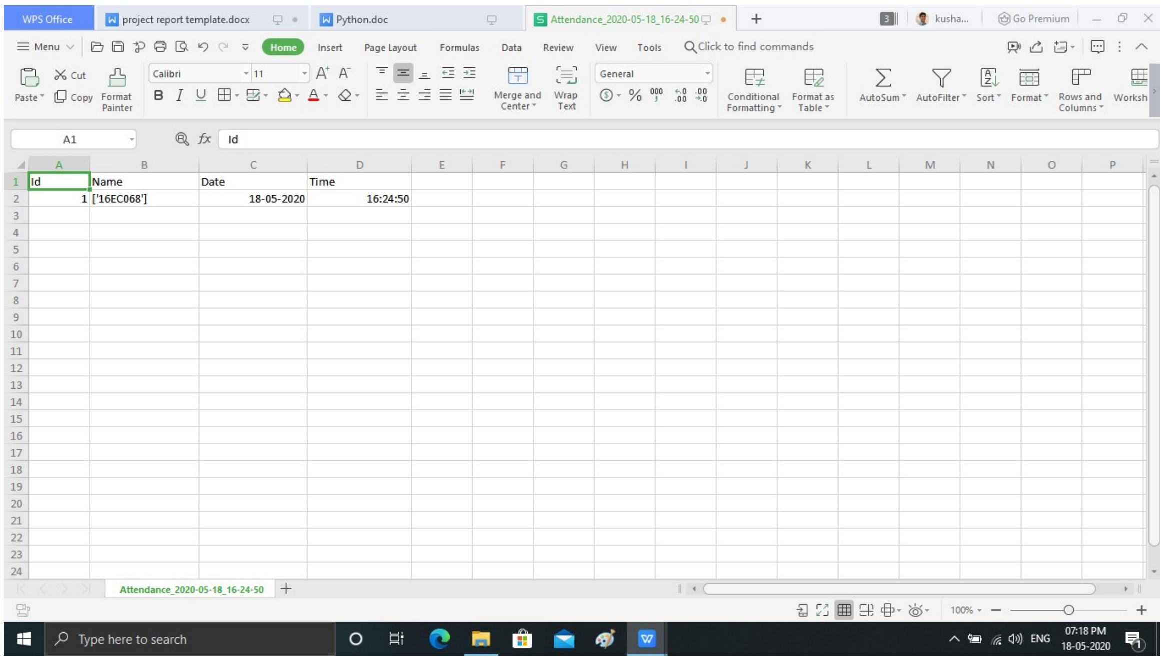


Figure.16 Face recognized

Step 4: Record the data of the person



Id	Name	Date	Time
1	['16EC068']	18-05-2020	16:24:50

Figure.17 Recorded data

The above observed results state that if a person is not wearing ID card then the person is identified and the details of that person is recorded into a excel sheet.

Chapter 8

CONCLUSION

A pre-trained SSD object detection model was tested for detecting ID Cards, both as-is and as fine-tuned with a manually tagged data set. Model testing and training was done using features provided by TensorFlow Object Detection API, a freely available framework for object detection. Following conclusions were drawn based on the results:

1) The pre-trained model was very accurate in testing set.

2) A fine-tuned model worked reasonably well.

3) Problem areas were moving object and object of same size.

4) A model trained with data from one background was able to detect in another. The overall model performance did not much improve by training the model with different frame rate.

5) The model performance evaluated as mAP improved when the model was trained from 17 400 to 200 000 timesteps. Optimal number of timesteps is likely to be around 100 000 timesteps.

6) Face recognition using LBPH is a trenmendous success.

8.1 Future work

1) A model with another type of meta architecture such as Faster R-CNN could be

better suited for detecting small objects.

2) Image pre-processing could improve detection results.

3) A more comprehensive testing protocol should be developed, and larger and

more variable datasets used for training and testing the model

REFERENCES

[1] Tran D (2017) How to train your own Object Detector with TensorFlow's Object Detector API.

[2] [2] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y & Guadarrama S (2016) Speed/accuracy trade-offs for modern convolutional

object detectors. arXiv preprint arXiv:1611.10012 .

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information

Processing Systems, pages 1097–1105, 2012.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings IEEE Conference onComputer Vision and Pattern Recognition, pages 1–9. IEEE, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778. IEEE, 2016.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015.

[8] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In Advances in Neural Information Processing Systems, pages 379–387, 2016.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European Conference on Computer Vision, pages 21–37. Springer, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255. IEEE, 2009.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll´ar, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In European Conference on Computer Vision, pages 740–755. Springer, 2014.

[12] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. OpenImages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://storage.googleapis.com/openimages/web/index.html, 2017.

[13] Fares Jalled and Ilia Voronkov. Object Detection Using Image Processing. arXiv preprint arXiv:1611.07791, 2016.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. Nature, 521(7553):436, 2015.

[15] Tom M. Mitchell. Machine Learning. McGraw-Hill Book Company, 1997.

[16] Michael A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015.

[17] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomForest. R News, 2(3):18–22, 2002.

[18] Robert D. Hof. Deep Learning: With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart. MIT Technology Review, 2018.

[19] Redmon J, Divvala S, Girshick R & Farhadi A (2016) You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. : 779-788.

[20] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C & Berg AC (2016) Ssd: Single shot multibox detector. European conference on computer vision. , Springer:

21-37.

[21] D. Zhang, S. Li, and D. Gatica-Perez. Real-time face detection using boosting learning in hierarchical feature spaces. In Proceedings of the International Conference on Pattern Recognition (ICPR), pages 411–414, Cambridge, UK, 2004.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed,Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European Conference on Computer Vision, pages 21–37. Springer, 2016.

[23]Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey

Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al.

Tensorflow: a system for large-scale machine learning. In OSDI, volume 16, pages

265–283, 2016.

[24] Ladislav Rampasek and Anna Goldenberg. Tensorflow: Biology's gateway to

deep learning? Cell Systems, 2(1):12–14, 2016.

[25] Peter Goldsborough. A tour of Tensorflow. arXiv preprint arXiv:1610.01178,

2016.

[26] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara,

Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, volume 4. IEEE, 2017.

[27] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on feature distributions,” Pattern Recognition, vol. 29, no. 1, pp. 51–59, 1996

Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA-2020)
DVD Part Number: CFP20N67-DVD; ISBN: 978-1-7281-5373-5

ID Card Detection with Facial Recognition using Tensorflow and OpenCV

Kushal M
Dept.of ECE
CMR Institute of Technology
Bengaluru, India

Kushal Kumar B V
Dept.of ECE
CMR Institute of Technology
Bengaluru, India

Charan Kumar M J
Dept.of ECE
CMR Institute of Technology
Bengaluru, India

Prof.Pappa M
Dept.of ECE
CMR Institute of Technology
Bengaluru, India

Abstract—The main goal of computer vision is to identify and recognize different objects of various size, shape and position. The major problems faced by the computer vision is the illumination and the viewpoint of the object, Concerning this and by following multiple studies recurring to deep learning with the use of Convolution Neural Networks on detecting and recognizing objects that showed a high level of accuracy and precision on these tasks. To facilitate object detection in a college environment, the proposed work identifies the presence of a person wearing an ID card using tensor flow object detection API, detects and recognizes the face using haar cascade method of OpenCV.

Keywords—ID card, Face detection, Face recognition, Tensor flow, OpenCV.

I. INTRODUCTION

Object detection is used to recognize certain objects like tree, humans, vehicles etc. The object detection belongs to computer vision and depends on many factors like size, shape, scale and illumination. The image processing field of computer vision can interact to recognize a specific part of the image. The image used for processing can be obtained from an image captured from a camera or a video frame. Computer vision is widely used in these days for many applications including [1] detecting moving objects.

Object detection is a broad area in the field of computer vision and Image processing. Face detection is one of the major research fields under object detection. The motivation behind face detection algorithms is to locate whether there are any faces in an image or not. In an image to recognize the face in it, first, have to detect[2] whether there is a face in the image or not. An image has to be generated which can be manipulated and used for various identification is important in face area determination. Once the face is detected, using the location of the detected face, the face recognition algorithm can be used to find the person in it. Over time face recognition have been used in many security [3]applications, face recognition based attendance systemetc.

This paper discusses how tensor flow can be used for object detection technique to find if the person is wearing an ID card or not. If not then it will proceed for face detection which will identify the presence of the face and also provide the coordinates of the location. Using these coordinates of the detected face, it recognizes the person in that video frame.

II. DIGITAL IMAGE DETECTION

A. Digital image:

An image captured is called the digital image or an RGB image. A sample digital image is shown in Figure 1. A digital image which is a 2D matrix consisting of intensity values as the elements of matrix can be obtained by sampling a 3D object.

Figure 1: Digital Image

The digital image is basically a two-dimensional representation of any object, it is a collection of RGB colors which contain different patterns and abstracts. An image can be classified into two categories 1. Still image 2.Moving image. A still image is the one which doesn't move to time, whereas a moving image is a collection of still images which changes to time (single frame of a video). [2]

B. Image detection

From the digital image captured certain pattern, shape, letter, object etc is need to be detected. An efficient architecture for computer vision based on deep neural network namely, codenamed Inception. This includes increasing the depth[5] and its range. Image net can be considered as a beneficial method for visualization applications such as Object recognition, Image classification and localization of the objects. Image detection deals with detecting the above-mentioned items in a given digital image. An example of image detection is shown in figure 2.

In the figure shown below, it is trained to identify the stop signboard in the digital image. The main problems faced by the digital image detection are the [4] changes in shape, size, scale etc. It also faces issues regarding the illumination of the image. The main function of the image detection is to identify if an object is present in that image or not, if yes then it has to provide the coordinates of that object.

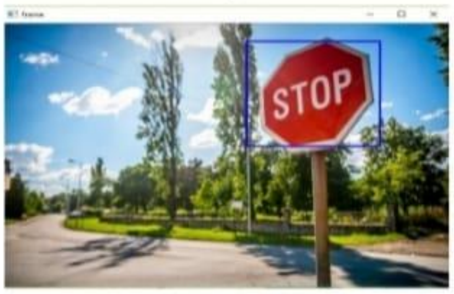


Figure 2: Image Detection

III. RELATED WORK

Tran D proposed [1] a method that had chosen images with large variations in scale, pose and lighting. He has labelled them manually with LabelImg. He has analysed by converting them into csv file and created TFRecord and the detector was trained based on the single class.

Ali Sharifara, Mohd Shafry, et al. [2], described a review of recent methods of face detection such as template matching, knowledge, appearance and feature-based were discussed and used neural networks along with Haar-like features algorithm. This has an advantage that it decreases the classifier disadvantages.

R. Samet and M. Tanriverdi [3], discussed mobile-based face recognition system for classroom attendance management was developed using Euclidean distance calculation. Distance calculation was carried out by considering Eigenfaces, Fisherfaces and LBP.

Tsang Ing Ren, Joao Paulo Magalhaes, et al.[4] described a template matching based face recognition for dynamic faces which has either horizontal or vertical movements. The author has used an algorithm to identify the face by distinguishing the skin region and the non-skin region. He has detected and identified using a neutral position and rotating position.

Christian Szegedy, Wei Liu, et al.[5], had used neural networks for computer vision. He has proposed Inception architecture that makes use of local sparse structure in convolutional vision network that leads to optimum results.

Jia Deng, Wei Dong, et al.,[6] has suggested that ImageNet can be used as a large-scale database for most of the applications such as Object recognition, Localization and Image Classification in the field of visual recognition.

Koresh, M.H.J.D,[7] has developed a framework based on capsule neural network which provides high reliability in sensing compared to convolutional neural networks.

IV. IMPLEMENTATION METHODOLOGY

A. Tensor flow Object Detection

This process first includes the data [8] collection, the collection of various images[9] that are to be identified or detected is called a dataset. There are a few publicly available datasets, but to make the system identify a customized object our own dataset have to be built. All the images are given an ID of what it is (image labeling). This dataset can be used to get the object detection[10] model using tensor flow which is pre-trained. In the beginning, the datasets are in XML format, which is then converted to CSV format and in turn to TFRecord format, which is then label mapped and trained to obtain the inference graph which is used for object detection.

Similarly to identify the ID card, the system is trained with the number of ID card images in various positions and background. Using this ID card dataset a Tensor flow object detection model is created and is shown in Figure 3.

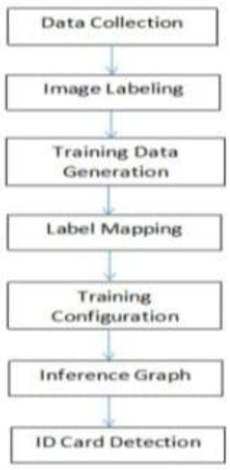


Figure 3: ID card detection flow

B. Face Detection and Recognition

This process is to find all the possible faces in the video frame. This procedure is only performed if the person is not wearing the ID. OpenCV hair cascade method is used to perform the concept of Cascade of Classifiers. The system is trained with a set of images which are to be detected [13]and with a set of images which are not to be detected and to generate an XML file.

The collected dataset (frontal face images of a different person for face detection) is trained and an XML file is obtained using the detection process as shown in Figure 4.

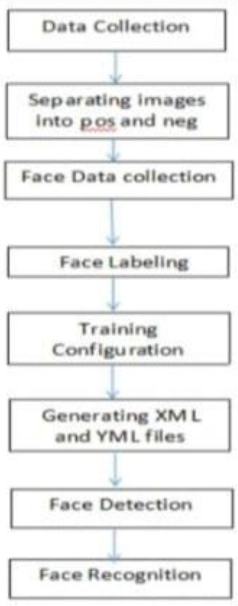


Figure 4: Steps involved in the face detection and identification method

For the recognition of the face, face data of a person who has to be recognized is collected. If more the number of images, the greater the efficiency, and here it is trained using 100 images. The face of the person is trained with an ID name. A YML file[12] is obtained after training. This file is used to recognize the person detected in the above part.

C. Framework

The outline of the proposed methodology is shown below in Figure 5.

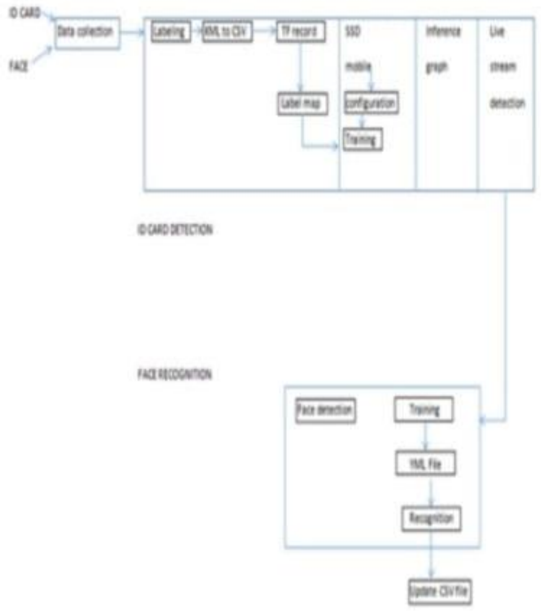


Figure 5: Framework

1. ID card detection:

In this process, if a person is wearing an ID card or not will be identified. This is done using a process called object detection using tensor flow. Here the system is trained with different ID card pictures and a trained inference graph is obtained which is used to identify the ID card in the reference image.

This process is achieved using the following steps:

- Data collection
- Image labeling
- Training data generation
- Label mapping
- Training configuration
- Inference graph
- ID card detection

2. Face Detection:

Face detection describes the process of determining whether there is a face in the image or not, and provide the coordinates of the face in the detected image[11] if it is detected. For this process, the system has to be trained with various face images. This process is performed using the haar cascade method. In this method two types of data are collected (i.e.) They are Positive images, which contain the images that have to be identified, and in this case, it is a face and Negative images which can contain any images apart from that have to be identified, that is anything else except for faces. The collected dataset is trained and an XML file is obtained

This process involves the following steps:

- Data collection
- Collecting positive images (images containing faces)
- Collecting negative images (images which do not have any faces in it)
- Training the system with the details of positive and negative images
- Generating an XML file
- Face detection

3. Face Recognition:

This process is to recognize the face which is detected as per the previous section. LBPH (local binary pattern histogram) is usually preferred face recognition and pattern recognition method in computer vision. It is appropriate for the feature because it describes the texture and structure of an image. In a grayscale image, a small part of the image is taken as a window of 3x3 pixels. It can also be represented as a 3x3 matrix containing the intensity levels.
The centre of the matrix is taken as the threshold, and a binary matrix is built using the threshold value, where the intensity value greater than the threshold is taken as 1 and the intensity values lesser are taken as 0. Using the binary value that is taken line wise is then converted into decimal value and set to the center of the matrix as shown below in figure 6.

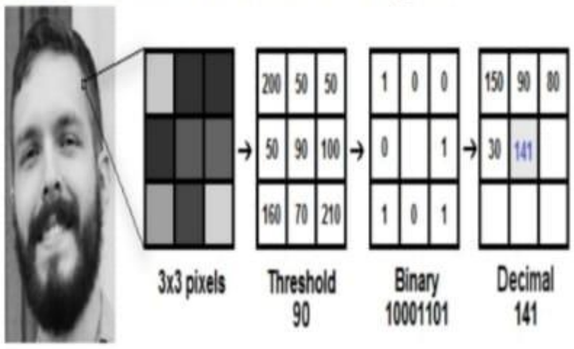


Figure 6: Local binary pattern

Once the above process is done the image is converted into grids. A histogram is formed for every grid and is combined to one histogram constituting all the grid histograms as shown below in figure 7.

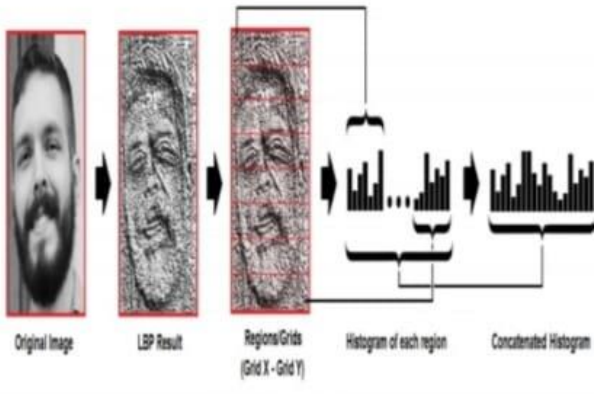


Figure 7: LBPH

The process is performed using the following steps:

- Collecting face data
- Labeling the face data
- Generating a YML file
- Face detection
- Face recognition

V. Result and Analysis

There are two cases in this system: case 1 is when the person is wearing an ID card and case 2 is when the person is not wearing an ID card. The results for both the cases are derived below.
Case 1: Person wearing an ID card
The proposed work is mainly designed to find out if the person is wearing an ID card or not and this is done by object detection using tensor flow. If the person is wearing an ID card, the system will mark the ID card with a green rectangle as shown in Figure 8.

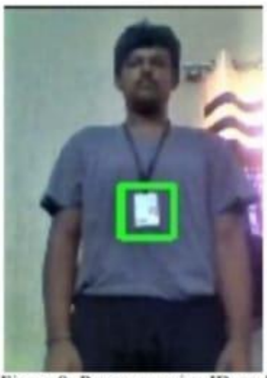


Figure 8: Person wearing ID card

Case 2: Person not wearing ID card.
The following steps are followed.
Step 1: ID card not detected.

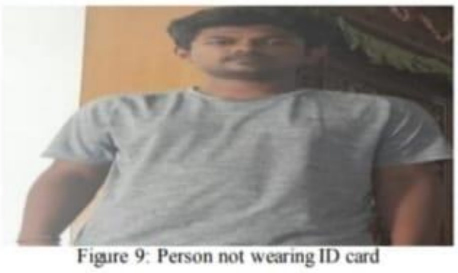


Figure 9: Person not wearing ID card

Step 2: Detect the face present in the image.

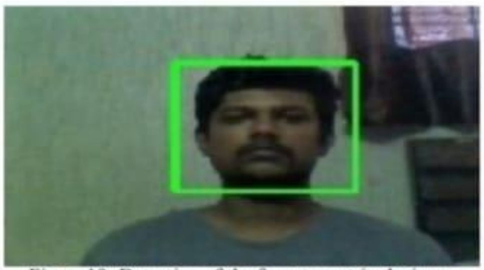


Figure 10: Detection of the face present in the image

Step 3: Identify the detected face in the image.

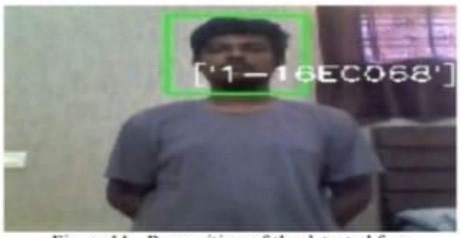
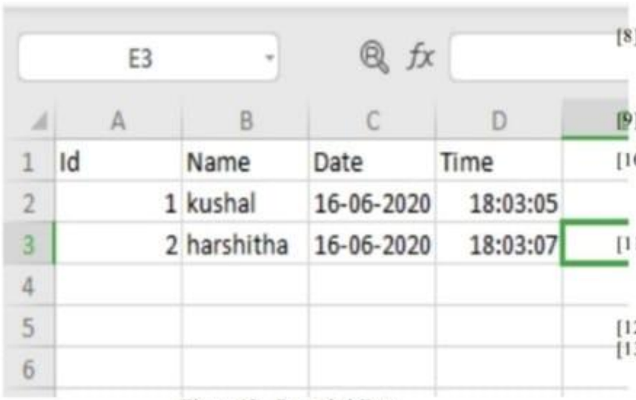


Figure 11: Recognition of the detected face

Step 4: Record the data of the person.



	A	B	C	D
1	Id	Name	Date	Time
2	1	kushal	16-06-2020	18:03:05
3	2	harshitha	16-06-2020	18:03:07
4				
5				
6				

Figure 12: Recorded Data

The above figure shows that if a person is wearing an ID card then the person is identified and the details of that person are recorded into an excel sheet.

VI. Conclusion

The Proposed model was tested for detecting ID Cards, faces and to recognize faces. Model testing and training were done using Tensor Flow Object Detection API with the help of features provided for ID card detection, Harrcascade classifier for face detection and LBPH method for face recognition was implemented. To conclude by observing the results:

1) The system was able to detect the presence of an ID card in the image.
2) It was able to detect the presence of faces and provide their co-ordinates.
3) Problem areas were moving object and object of the same size.
4) Face recognition using LBPH is a tremendous success.

REFERENCES

[1] "No Titl." https://towardsdatascience.com/ how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9.

[2] A. Sharifara, M. S. Mohd Rahim, and Y. Anisi, "A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection," *Proc. - 2014 Int. Symp. Biometrics Secur. Technol. ISBAST 2014*, pp. 73–78, 2015, doi: 10.1109/ISBAST.2014.7013097.

[3] R. Samet and M. Tanriverdi, "Face recognition-based mobile automatic classroom attendance management system," *Proc. - 2017 Int. Conf. Cyberworlds, CW 2017 - Coop. with Eurographics Assoc. Int. Fed. Inf. Process. ACM SIGGRAPH*, vol. 2017-Janua, pp. 253–256, 2017, doi: 10.1109/CW.2017.34.

[4] S. R. Christian Szegedy1, Wei Liu2, Yangqing Jia1, Pierre Sermanet1 and A. R. Dragomir Anguelov1, Dumitru Erhan1, Vincent Vanhoucke1, "Going Deeper with Convolutions," 2015.

[5] I. R. Tsang, J. P. Magalhaes, and G. D. C. Cavalcanti, "Combined AdaBoost and gradientfaces for face detection under illumination problems," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, pp. 2354–2358, 2012, doi: 10.1109/ICSMC.2012.6378094.

[6] K. L. and L. F.-F. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, "ImageNet: A Large-Scale Hierarchical Image Database," 2009.

[7] J. D. K. H, "Computer Vision Based Traffic Sign Sensing for Smart Transport," *J. Innov. Image Process.*, vol. 1, no. 01, pp. 11–19, 2019, doi: 10.36548/jiip.2019.1.002.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[9] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[10] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[12] C. Grosan and A. Abraham, *Machine Learning*, vol. 17. 2011.

[13] F. Jalled and I. Voronkov, "Object Detection using Image Processing," pp. 1–6, 2016, [Online]. Available: http://arxiv.org/abs/1611.07791.