

Fruits and Vegetables Detection For Visually Impaired Person

Muntasir Feroz
2015-1-60-058

Noor Nahian
2015-1-60-096

Kamrun Nahar Kona
2015-2-60-008

A thesis submitted in partial fulfillment of requirements for the
degree of Bachelor of Science and Engineering



Department of Computer Science and Engineering
East West University
Dhaka-1212, Bangladesh
September, 2019

Declaration

We, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by us under the supervision of Dr. Mohammad Salah Uddin, Assistant Professor, Department of Computer Science and Engineering, East West University. We also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

Signature

.....
(Dr. Mohammad Salah Uddin)
Supervisor

.....
(Muntasir Feroz)
(ID: 2015-1-60-058)

.....
(Noor Nahian)
(ID: 2015-1-60-096)

.....
(Kamrun Nahar Kona)
(ID: 2015-2-60-008)

Letter of acceptance

This project entitled "*Fruits and Vegetables Detection For Visually Impaired Person*" submitted by Muntasir Feroz, ID: 2015- 1-60-058, Noor Nahian , ID: 2015-1-60-096 and Kamrun Nahar Kona , ID: 2015-2-60-008 to the Computer Science and Engineering Department, East West University, Dhaka-1212, Bangladesh is accepted as satisfactory for partial fulfillment of requirements for the Award of Degree of Bachelors of Science (B. Sc.) in Computer Science and Engineering on December, 2019.

Supervisor

.....

(Dr. Mohammad Salah Uddin)

Assistant Professor,

Department of Computer Science and Engineering,

East West University

Chairperson

.....

(Dr. Taskeed Jabid)

Chairperson and Associate Professor,

Department of Computer Science and Engineering,

East West University

Abstract

Fruits and Vegetables Detection For Visually Impaired Person plays an important role in Application Systems. It helps the customers to identify their desired fruits and vegetables not only by image but also with identifying sound. The main purpose of this mobile application system is to help blind people to identify fruits and vegetables so that they can purchase their food on their own. It can be also used by robots and in the conveyor belt of factories. However, convolutional neural networks have proved to be potentially more effective. In this thesis, we present a convolutional neural network trained to classify and detect fruits and vegetables from multiple angles.

Acknowledgement

First of all, we would like to express our deepest gratitude to the almighty Allah for his blessings on us. Next, our special thanks go to our supervisor, **Dr. Mohammad Salah Uddin** who gave us this opportunity, initiated us into the field of “Computer Vision and Deep Learning”, and without whom this work would not have been possible. His encouragements, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of our BSC study were simply appreciating and essential. His ability to muddle us enough to finally answer our own question correctly is something valuable what we have learned and we would try to emulate, if ever we get the opportunity. There are numerous other people too who have shown me their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our hearts and hope to find a more appropriate place to acknowledge them in the future.

Muntasir Feroz

September, 2019

Noor Nahian

September,2019

Kamrun Nahar Kona

September, 2019

List of Figures:

- **Figure 2.1 : Array of RGB Matrix.**
- **Figure 2.2 : Neural network with many convolutional layers.**
- **Figure 2.3 : Image matrix multiplies kernel or filter matrix.**
- **Figure 2.4 : Image matrix multiplies kernel or filter matrix.**
- **Figure 2.5 : 3 x 3 Output matrix.**
- **Figure 2.6 : Some common filters.**
- **Figure 2.7 : Max Pooling.**
- **Figure 2.8 : After pooling layer, flattened as FC layer.**
- **Figure 2.9 : complete CNN architecture .**
- **Figure 2.10 : SSD Flowchart.**
- **Figure 2.11 : VGG based SSD Architecture**
- **Figure 2.12 : Selective Search.**
- **Figure 2.12 : VGG based SSD Architecture**
- **Figure 2.13 : An example of regular convolution³**
- **Figure 2.14 : Two different steps of depth wise separable convolution⁴**
- **Figure 3.1 : Deep learning based model.**
- **Figure 4.1 : Decline of total loss when fine-tuning SSD MobileNet**
- **Figure 5.1 : Fruits and vegetables images with predicted boundary box.**
- **Figure 5.2 :Accuracy graph**

List of Equations:

1. $accuracy = \frac{TP}{all\ detection} \times 100$

Contents

Declaration	i
Letter of acceptance	ii
Abstract	iii
Acknowledgement	iv
List of Figures	v
List of Equations	vi
1. Introduction	1
1.1 Introduction	1
1.2 Objectives	2
1.3 Contribution	3
1.4 Roadmap	3
2. Background study	4
2.1 Convolutional neural network	4
2.1.1 Convolution layer	5
2.1.2 Pooling layer	8
2.1.3 Fully connected layer	9
2.2 SSD	10
2.3 Selective Search	12
2.3.1 SSD MobileNet	12
3.Related work	15
3.1 Introduction	15
3.2 Object detection	15
3.3 Object classification	16
3.4 Different models	17
3.4.1 Feature based model	17
3.4.2 Deep learning model	19
4.Methodology	21
4.1 Introduction	21

4.2 Data Collection and pre-processing	21
4.3 Tensorflow object detection API for fine-tuned model	22
4.3.1 Object Detection with SSD MobileNet	23
4.3.2 Object Detection with Voice identification Android App for Visually Impaired Person	29
5. Result	30
6. Conclusion	32
6.1 Summary	32
6.2 Future Work	32
References	33

Chapter 1

Introduction

1.1 Introduction

Fruits and Vegetable classification is a difficult and important task in supermarkets since it is necessary for the cashier to know the categories of particular fruit and vegetable in order to determine its price [17]. The packaging of the vegetable like other food products has barcodes. The objective of this paper is to propose a novel fruits and vegetable classification system based on computer vision, with the aim of solving this type of problem [16]. First, we use merely a cell phone camera, getting rid of other complicated hardware. Second, the proposed classifier is expected to recognize as many types of vegetables as possible. Different fruits and vegetable samples like Cucumbers, Tomatoes, Onions, Carrots, Apple, Orange, and Guava etc are considered in the work. Traders have warehouses where different varieties of fruits and vegetable technology and make the training parameters greatly reduced compared to the neural network [13]. It also has a certain degree of translation, rotation and distortion and fruits and vegetables are stored. Most fruits and vegetables dealers will sort the fruits and vegetables manually which results in high cost, subjectivity, tediousness, and inconsistency associated with manual sorting [13]. Therefore, different fruits and vegetable varieties easily get mixed up during harvesting, storage, and marketing.

Classification is a fundamental research work in the field of Agriculture and Botany. Up to now, it has been found that there are hundreds of thousands of species of vegetables and fruits [4]. People will get confused because they don't know the species of vegetables and fruits. Therefore, the design of fruits and vegetable classifier will also bring ease to people's lives [1]. There are some challenges in fruits and vegetable classification, the background of fruits and vegetable image is complex, there is similarity between the different species of fruits and vegetables, so we cannot just rely on a single feature, such as color, shape or texture to distinguish the species of fruits and vegetables [11] [8], and the same species of fruits and vegetables will be different because of the shape, scale, viewpoint and so on [9]. The idea is based on Martin Gerner's Hand-written digit recognition using Tensorflow for Poets [20]. A similar procedure is followed for developing this android based system. The idea is to use computer vision, image processing and convolutional neural networks [12]. The convolutional neural network is an efficient recognition method which has been developed in recent years [2]. This network avoids the complex preprocessing of the image, and people can input the original image directly [7]. It uses the local receptive field, weights sharing and pooling invariance of the image. It has made great progress in

the field of image classification [9]. TensorFlow [1] is the second generation of artificial intelligence learning system developed by Google, which supports the convolution neural network (CNN), recurrent neural network (RNN) and other depth of the neural network model, which can be used in speech recognition, image recognition and so on many machines deep learning field [20]. In this paper, we use the transfer learning technique to retrain the Inception-v3 [3] model of TensorFlow [1] on fruit and vegetable category datasets [13] [11]. We implemented an effective fruits and vegetable classification model using a short training time and achieve higher accuracy. Transfer learning is a new machine learning method that can use the existing knowledge learned from one environment and solve the other new problem which is different but has some relation to the old problem. For example, we can apply the knowledge learned from the motorcycle problem to the study of bike problems [16]. Compared with the traditional neural network, it only needs to use a 784 small amount of data to train the model, and achieve high accuracy with a short training time [14].

This presents a unified approach that can combine many features and classifiers [3]. The training method implemented is efficient compared to traditional, where all features are simply concatenated and fed independently to each classification algorithm. We expect that this solution will endure beyond the problem solved in this paper.

1.2 Objectives

The goal of our thesis is to develop a convolutional neural network (CNN) which can detect fruits and vegetable with voice feedback.

The objectives are given bellow more precisely:

- Implement a classifier that is able to predict correct image classes: fruits and vegetable with voice feedback.
- Implement a fruits and vegetable detector that can help the blind people to select correct fruits and vegetable.
- Using Tensorflow API.
- Improve or reduce the accuracy by increasing and decreasing of threshold value.

In this thesis, the detection is limited for arbitrary image.

1.3 Contribution

The contributions include a SSD for fruits and vegetable detection and classification. In addition to that, investigating the effect of SSD on input data. To achieve that the following steps have to be done:

- Finding the fruits and vegetables datasets.
- Applying preprocessing techniques to the collected data. This step includes the experimental pre-processing.
- Developing SSD architecture suitable for the dataset and the goal of this thesis. Experimenting with different hyper parameters is essential to get a good model.
- Training the SSD to do fruits and vegetable detection.
- Testing the final solution.

1.4 Roadmap

Chapter 1:

This chapter represents the motivation of our work, the objectives and the contribution that we have made.

Chapter 2:

Introducing background study that use SSD for object detection as well as fruits and vegetable detection.

Chapter 3:

Introducing related work that uses SDD for object detection as well as fruits and vegetable detection.

Chapter 4:

The methodology of the experiments is presented. This chapter includes datasets, pre-processing, training of the SSD and transfer learning.

Chapter 5:

Results of the experiment presented. This chapter also contains the accuracy of the using model on the collected dataset as well as the evaluation of the using model.

Chapter 6:

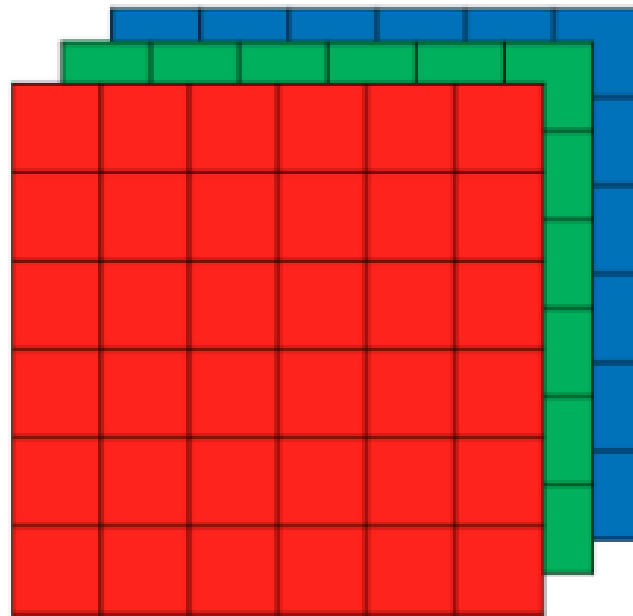
We conclude the thesis work and future work by using the SSD models.

Chapter 2

Background Study

2.1 Convolutional neural network

In neural networks, Convolutional neural network is one of the most popular algorithms to do images recognition, images classifications. Objects detections, recognition faceetc. are those areas where CNNs are widely used. CNN image classifications takes an input image then process the and classify it under certain categories (E.g. Apple, Orange, Banana, Potato, Cucumber). Computers see an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see $h \times w \times d$ (h = Height, w = Width, d = Dimension). E.g. An image of $6 \times 6 \times 3$ array of matrix of RGB (3 refers to RGB values) and an image of $4 \times 4 \times 1$ array of matrix of gray scale image.



6 x 6 x 3

Figure 2.1: Array of RGB Matrix

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

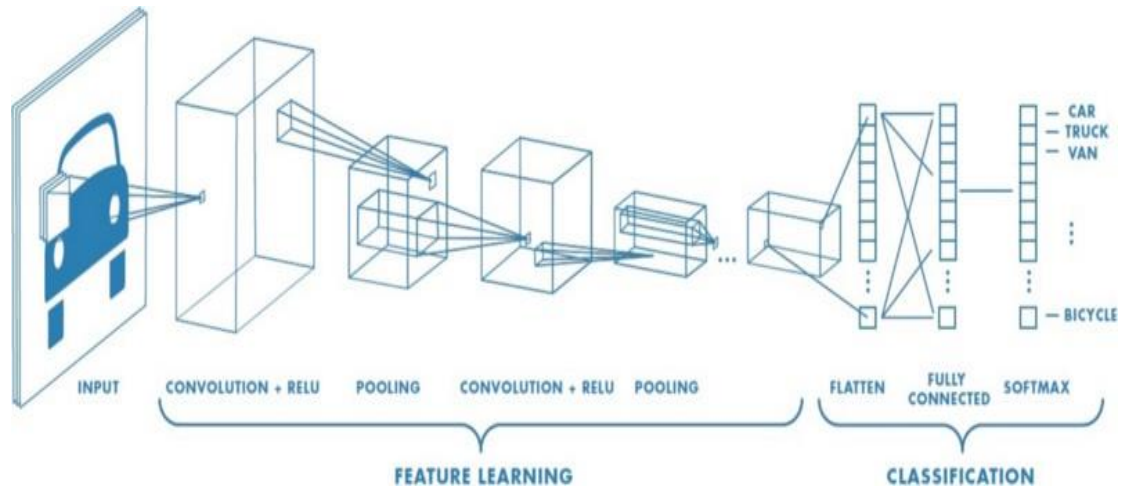


Figure 2.2 : Neural network with many convolutional layers

2.1.1 Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension **($h \times w \times d$)**
- A filter (**$f_h \times f_w \times d$**)
- Outputs a volume dimension **($h - f_h + 1$) \times ($w - f_w + 1$) \times 1**

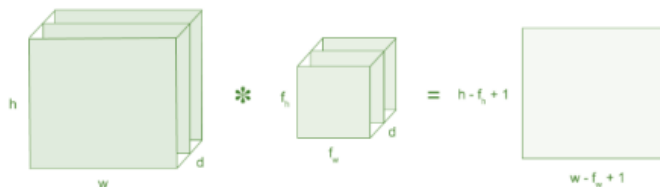


Figure 2.3: Image matrix multiplies kernel or filter matrix

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

*

1	0	1
0	1	0
1	0	1

5 x 5 – Image Matrix
3 x 3 – Filter Matrix

Figure 2.4: Image matrix multiplies kernel or filter matrix

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “**Feature Map**” as output shown in below

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Image
Convolved Feature

Figure 2.5: 3 x 3 Output matrix

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels).








Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 2.6: Some common filters

2.1.2 Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called sub-sampling or down-sampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

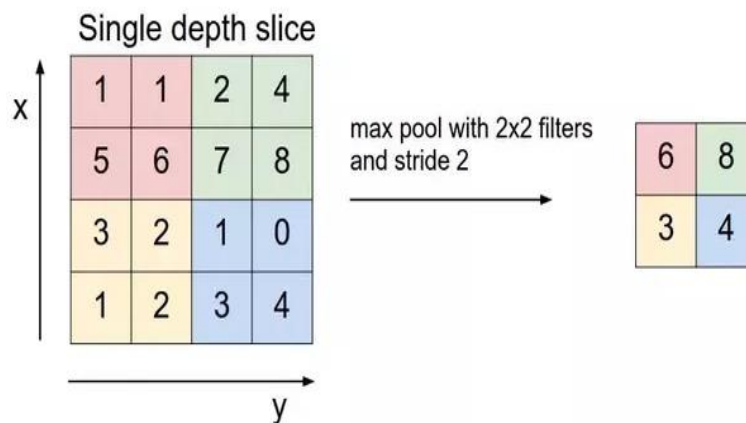


Figure 2.7: Max Pooling

2.1.3 Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

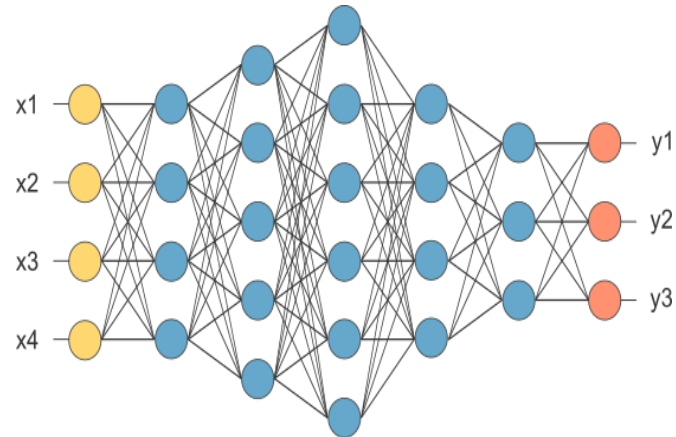


Figure 2.8 : After pooling layer, flattened as FC layer

In the above diagram, feature map matrix will be converted as vector (x_1, x_2, x_3, \dots). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.

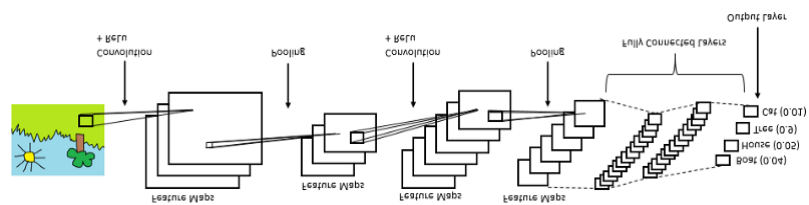


Figure 2.9 : Complete CNN architecture

2.2 SSD

By using SSD (Single Shot Detector) , we only need to take one single shot to detect multiple objects within the image, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches. SSD300 achieves 74.3% mAP at 59 FPS while SSD500 achieves 76.9% mAP at 22 FPS, which outperforms Faster R-CNN (73.2% MAP at 7 FPS) and YOLOV1 (63.4% mAP at 45 FPS).

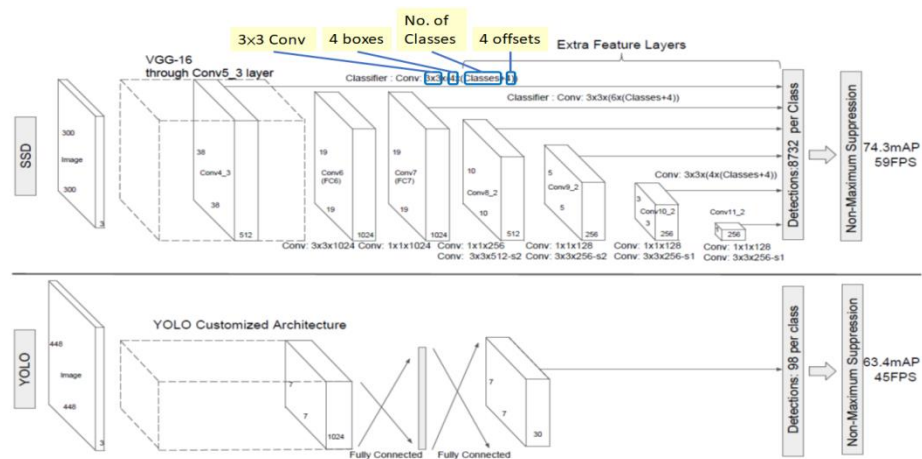


Figure 2.10:SSD Flowchart

To have more accurate detection, different layers of feature maps are also going through a small 3×3 convolution for object detection as shown above.

Say for example, at Conv4_3, it is of size $38 \times 38 \times 512$. 3×3 conv is applied. And there are 4 bounding boxes and each bounding box will have (classes + 4) outputs. Thus, at Conv4_3, the output is $38 \times 38 \times 4 \times (c+4)$. Suppose there are 20 object classes plus one background class, the output is $38 \times 38 \times 4 \times (21+4) = 144,400$. In terms of number of bounding boxes, there are $38 \times 38 \times 4 = 5776$ bounding boxes.

Similarly for other conv layers:

Conv7: $19 \times 19 \times 6 = 2166$ boxes (6 boxes for each location)

Conv8_2: $10 \times 10 \times 6 = 600$ boxes (6 boxes for each location)

Conv9_2: $5 \times 5 \times 6 = 150$ boxes (6 boxes for each location)

Conv10_2: $3 \times 3 \times 4 = 36$ boxes (4 boxes for each location)

Conv11_2: $1 \times 1 \times 4 = 4$ boxes (4 boxes for each location)

If we sum them up, we got $5776 + 2166 + 600 + 150 + 36 + 4 = 8732$ boxes in total.

If we remember YOLO, there are 7×7 locations at the end with 2 bounding boxes for each location. YOLO only got $7 \times 7 \times 2 = 98$ boxes. Hence, SSD has 8732

bounding boxes which is more than that of YOLO. A typical CNN network gradually shrinks the feature map size and increase the depth as it goes to the deeper layers. The deep layers cover larger receptive fields and construct more abstract representation, while the shallow layers cover smaller receptive fields. For more information of receptive field, check this out. By utilizing this information, we can use shallow layers to predict small objects and deeper layers to predict big objects, as small objects don't need bigger receptive fields and bigger receptive fields can be confusing for small objects.

The following chart shows the architecture of SSD using VGG net as the base net. The middle column shows the feature map sets the net generates from different layers. For example the first feature map set is generated from VGG net layer 23, and have a size of 38×38 and depth of 512. Every point in the 38×38 feature map covers a part of the image, and the 512 channels can be the features for every point. By using the features in the 512 channels, we can do image classification to predict the label and regression to predict the bounding box for small objects on very point. The second feature map set has a size of 19×19 , which can be used for slightly larger objects, as the points of the features cover bigger receptive fields. Down to the last layer, there is only one point in the feature map set, which is ideal for big objects.

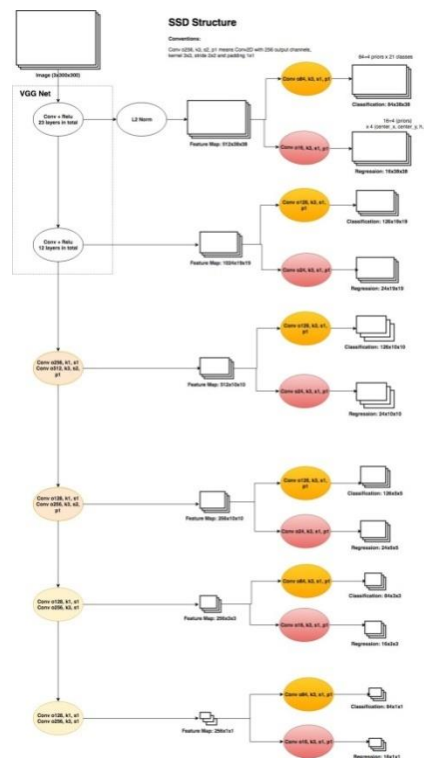


Figure 2.11: VGG based SSD Architecture

2.3 Selective search

1. Color similarities, texture similarities, region size, and region filling are used as non-object-based segmentation. Therefore we obtain many small segmented areas as shown at the bottom left of the image above.
2. Then, bottom-up approach is used that small segmented areas are merged together to form larger segmented areas.
3. Thus, about 2K region proposals (bounding box candidates) are generated as shown in the image.

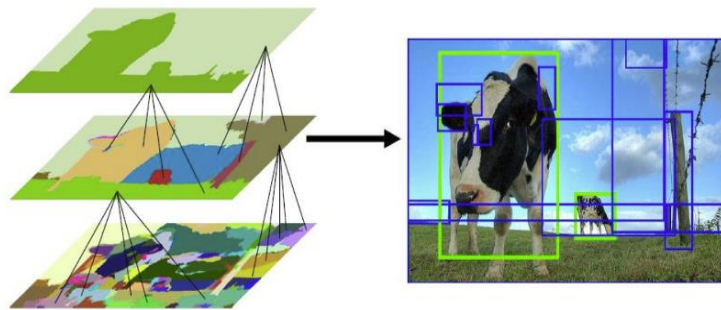


Fig2.12:Selective Search

2.3.1 SSD MOBILENET

SSD(Single Shot Detector) uses mobile-net as it's feature extractor. In order to increase the speed. Mobile-net is a lightweight convolutional neural nets specially using separable convolution in-order to reduce parameters. So separable convolution can lose information because of the channel wise convolution. So when quantifying a graph according to TF implementation it makes 16 bits ops and weights to 8 bits. This will work really well and almost lossless in terms of accuracy for a heavy model like inception, resnet etc. But with the lightness and simplicity of SSD with mobile-net it really can make an accuracy loss. So mobile-net is an efficient convolution neural network for mobile vision applications.

The state-of-the-art architectures described above are quite deep with up-to 152 layers and therefore require a considerable amount of memory and inference time. Also, most of the CNN needs high-end GPU devices for their training and inference. Moreover, as the network gets deeper and deeper, it demands more and more memory and computation, limiting their

operation on even fewer devices. Due to the above reasons, it becomes difficult to run deep network models directly on mobile devices, which is an extensive market nowadays [29, 45]. The drawbacks of the deeper networks motivated the researchers to search for solutions appropriate for mobile and embedded vision applications.

A group of researchers at Google proposed MobileNet, a family of architectures which encapsulates the CNN that runs efficiently on mobile devices. The authors claim that the accuracy of MobileNet is comparable to VGG16 although it is 32 times smaller than VGG16. The principle idea behind MobileNet is that it uses depth wise separable convolution to build quite small, low-latency deep neural networks.

In standard CNN, all the convolutional layers deploy regular convolution. Regular convolution means filtering all the input channels and combining them into a single output channel in one step as shown in Figure 2.8 below. However, in MobileNet architecture, the first layer still uses standard convolution while all the layers use depth wise separable convolution

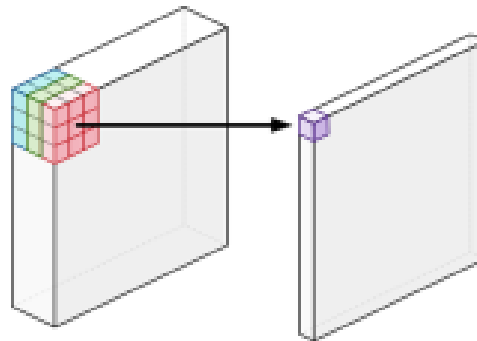


Figure 2.13: An example of regular convolution

The depth wise separable convolution is factorized into two different operations of convolution: a depth wise convolution and a point wise convolution. Unlike regular CNN, MobileNet filters the input channels in depth wise convolution and then combines them in point wise convolution step. The Figure 2.9 below shows two different steps of the depth wise convolution and the point wise convolution.

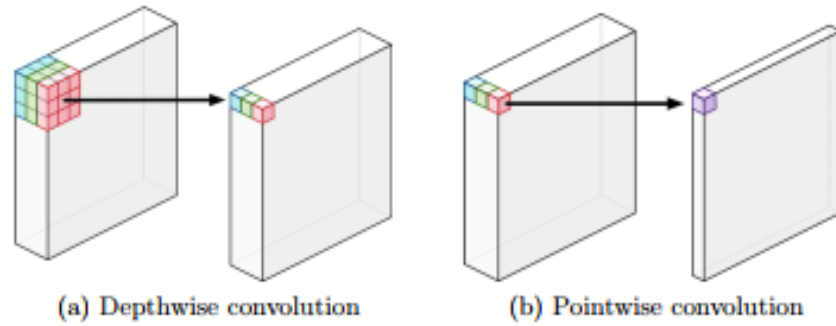


Figure 2.14: Two different steps of depth wise separable convolution4

The factorization process described above makes it possible to reduce the computation and size of the model substantially. The full MobileNet version 1 (v1) network has 30 layers. Though the size of MobileNet is pretty small as compared to other deep networks, it further can be shrunk by tuning hyper parameters such as width multiplier and resolution multiplier. The hyper parameter tuning makes the networks smaller, and faster but at the expense of prediction accuracy.

Chapter 3

Related work

3.1 Introduction

Several approach has been done to detect car in images HOG, SURF, SIFT and SVM CNN are the common features that has been used to detect car. In modern literature there are neural networks especially convolutional neural networks have been used to car detection.

3.2 Object detection

In 2013 Girshick et al. [9] introduced RCNN, a CNN with region proposals. In their paper, the authors wish to show that CNN is capable of achieving better results than methods using low-level features such as HOG. Their object detection pipeline consists of three parts. First of all, RCNN uses selective search to create region proposals, i.e. various regions that might include some object. Next, 4096-dimensional feature vectors are extracted from each detected region using AlexNet. The only change in the architecture is the number of units in the classification layer. Afterward, each feature vector is classified by using SVMs where each SVM has been trained for a specific class. On the ILSVRC2013 detection dataset, RCNN scored better than over feat.

Sermanet et al. come up with a neural network that can be used for three computer vision tasks: object classification, localization and detection. In addition to that, a feature extractor, called overfeat, is released. The authors aim to prove that CNNs are capable of doing more than classification. AlexNet is used as the base network for classifications. The modifications done to AlexNet include non-overlapping pooling, smaller stride size in the first two layers and omitting contrast normalization. For the localization task, the fully-connected layers of the classification model are changed to a regression network. The regression network is trained to output four bounding box coordinates. The classifier and the regression network are run simultaneously. Then, their results are merged to get the final predictions. The detection task is similar to the localization, but background class is added for images that do not contain any objects.

YOLO [1] is one of most recent object detection approaches. Some previous works use classifiers to perform object detection. In contrast, the proposed method predicts bounding box coordinates along with class score within the same neural network.

Unlike region proposal-based networks YOLO uses the whole image instead of separate regions to make decisions. The network architecture is based on GoogleLeNet where inception modules are replaced by reduction layers. Twenty convolutional layers are first pre-trained and then another four convolutional and two fully-connected layers are added to the model. Each input image is divided into a grid. Each grid cell predicts bounding box coordinates a bounding box confidence and a class probability.

Ming Liang and Xiaolin Hu [12] propose a recurrent CNN. This solution is inspired by recurrent connections in recurrent neural networks. The authors develop a recurrent convolutional layer that is used instead of regular convolutional layers in the proposed model. Only the first convolutional layer of the network is not recurrent. The base network and training process were adapted from AlexNet. The depth of the network is increased by adding recurrent connections.

The concept is to attempt to model a child's learning process, when related to predicting what new objects are like based on previously learned objects. A object is defined for this project as something like a ice cream cone or a pizza. It's something that consists of color and a specific set of environmental traits (hot/cold, luminous, etc.). This experiment takes in those traits via imagery and environment sensor data to define what an object is like. That data is then used to forma model of relationships of similar objects that can be later compared to a new object. When a new object is discovered the prediction process would try to fit it to an already learned object. Even if the new object isn't predicted correctly, there is still a valid path in its process to learn that new objects characteristics. If that path is taken, the algorithm performs a confirmation step where a correction could be made to then correctly learn that new object's characteristics. This concept is relying on using multiple sensor data sources to decrease error in the learning/predicting algorithm, but may still run into cases where the noise is too high to get an understanding of what is being analyzed. Understanding this noise is going to be part of the experimentation process and may limit the initial project inputs to controlled background colors and images that appeal to the available image detection algorithms.

3.3 Object classification

In 2012 Krizhevsky et al. [7] make a breakthrough in large dataset image classification by introducing AlexNet. While there are several others machine learning methods used at the competition, the authors prove that CNNs are capable of managing millions of images and achieve high results. The key to final results is the

depth of AlexNet. There is no preprocessing done on the dataset except for subtracting the mean. The proposed network consists of five convolutional and three fully-connected layers. The authors choose ReLU for the activation function since it drastically improved the speed of the training. Local response normalization is applied after the activation in first two layers. Moreover, overlapping pooling helped to reduce overfitting during the training. The final layer is a softmax layer with 1000 units. The training is done on two GPUs where each GPU has access to different layers. Data augmentation and dropout are used to reduce overfitting of the network.

In 2014 [13] another successful CNN architecture was introduced – VGGNet. With this neural network, the authors won the first and the second places in the classification and localization tasks of the ILSVRC'14 competition. The base of the network is inspired by AlexNet. Similarly to AlexNet, the only preprocessing was mean extraction. The authors experimented with different network depths to understand how it affects the final results. One of the differences from previous architectures is the use of smaller kernel size 3×3 in all convolutional layers. In addition to that, the CNN contains multiple sequential convolutional layers without pooling layer in between. The authors tested six different configurations of different depth. The best results were achieved when the network consisted of 16 and 19 layers.

CNN-RNN [14] is a convolutional neural network combined with a recurrent neural network that is used for multi-label classification. The goal of multi-label classification is to predict the labels of multiple objects in the same image. The authors observed that adding RNN to the original network increased the accuracy. Long short term memory units are used as recurrent units of this network. The CNN part of the network is based on VGGNet and it is used to collect semantic information from images. The RNN part deals with the relationship of images and labels.

3.4 Different Models

3.4.1 Feature Based Model

Car detection and classification has various solutions using descriptors like SIFT, SURF and HOG. Compared to CNNs the features extracted by previously mentioned methods are rather low-level. Typically a support vector machine (SVM) is used as the classifier. Typically a support vector machine (SVM) is used as the classifier.

Feature based recognition is quite different from a variety of object recognition algorithms. These algorithms rely on a single type of feature: an edge. Because the world is full of edges that look roughly the same, the set of edges extracted from an image cannot be directly compared to the set of edges extracted from a model object. Before a comparison can be made one must first discover the mapping, called a correspondence from image edges to model edges. There are many approaches to finding a correspondence, but in every case significant computation is required. FBR attempts to finesse the problem of finding a correspondence. Instead it constructs representations of the image and the model that are directly comparable. As a result recognition can be viewed as a classification problem. FBR proceeds by computing a number of properties of the input image and combining them into a feature vector. An object model is a set of feature vectors associated with a set of representative images of the object. A novel image is classified by computing the feature vector of the image and comparing it directly to the model vectors.

In the authors use a combination of SIFT and bag-of-words and perform the classification of vehicle make and model using SVM. Another approach first subtracts the background around the vehicle, then the image is segmented and SIFT is used to extract the features [2]. The authors perform feature matching to evaluate the results. Ma and Grimson[4] "adopt SIFT, with several key modifications tuned to car classification, as descriptors for edge points".

circle make and model recognition is performed using a modified SURF algorithm – a symmetrical SURF descriptor [5]. The idea of this modification is to identify whether the input contains symmetrical objects. The proposed method works well with real-time data and does not need background subtraction.

X. Li and X. Guo[6] developed a forward vehicle detection system that uses HOG descriptor and SVM. Their idea relies on detecting shadows underneath vehicles to distinguish between vehicles and background. Moreover, this method proved to perform well in different lighting conditions. Similarly to the previous research, Sivaraman and Trivedi[7] take advantage of shadows underneath vehicles. The features are extracted using HOG and the proposed symmetrical HOG that detects symmetrical features. In their research Feng Han et. al[8] use HOG together with SVM to detect people and vehicles from different viewpoints. Firstly, the authors use

stereo cue to detect either person or vehicle candidates. Then, the classification using HOG features is done with SVM. For each viewpoint, the authors develop separate classifiers that are applied simultaneously to get a joined result. Pablo Negriet. al[9] utilize HOG and Haar-like features in their vehicle detection research. In fact, the authors experiment with merging these descriptors and achieve accurate final results.

3.4.2 Deep learning based model

In present day deep learning is very popular in computer vision. Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks [10] and recurrent neural networks [10] have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts. In[12] the authors use CNN with low resolution video frames to detect and classify vehicles. The pre-processing stage includes re-sizing each frame and adjusting the contrast with histogram equalization. The proposed CNN architecture detects both high level and low-level features. The authors vary the number of filters and filter sizes as well as the number of hidden layers. In the paper by Heikki Huttunen et al.[13]a simple CNN outperforms the SIFT and SVM method when applied to vehicle classification.

Transfer learning is another approach used for vehicle detection and classification. It is a method where an already pre-trained model is used as a starting point or as a feature extractor. If needed the existing model can be fine-tuned further. In [14] the authors use fine-tuned YOLO [11] for vehicle detection and fine-tuned AlexNet model for vehicle classification. The authors additionally use AlexNet as feature extractor and perform classification on extracted features using linear SVM.

JorgeE.Espinosaetal.[15]comparetheperformanceofAlexNetandFasterRCNN [16]. Furthermore, Faster RCNN was fine-tuned and modified by Quarfu Fan et al. [17] to perform better on vehicle detection. In [18] Fast

RCNN[19], the predecessor of Faster RCNN, has been applied to vehicle detection, although the authors simplified it for their task.

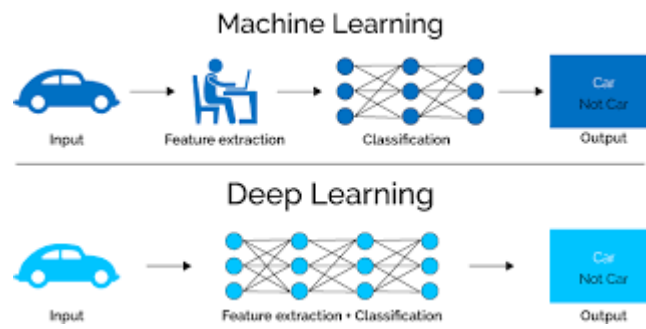


Figure 3.1 : Deep learning based model

Chapter 4

Methodology

4.1 Introduction:

One of the main objectives of our thesis is to train the best object detector with our fruits and vegetables dataset. 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 Tensorflow pre-trained models of object detection. Lastly, we describe the evaluation.

4.2 Data Collection and pre-processing

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 numbers 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 but in our case we could a complete dataset of both fruits and vegetables so we merged different dataset and collected some images from internet. We merged fruit-images-for-object-detection from kaggle with the collected images from internet.

Selection and Annotation of Datasets: We selected and annotated around 6004 examples altogether. There are total 24 fruit classes and 16 vegetable classes each having about 100 instances. 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, 1 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 files.

Create training and test dataset: After completing the annotation process, we divided our fruits and vegetables dataset into two sets: training dataset and test

dataset. Finally, we arrived at one training dataset with 3808 examples, one train dataset with 2188 examples.

Create TFRecords: 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. The python command to convert a CSV file into a TFRecordfile is given below:

```
# Convert a training dataset csv file into training TFRecord
python generate_tfrecords.py
--csv_input=path/to/train_labels.csv
--output_path=path/to/train.record
```

Tensorflow models are pre-trained for multiple datasets such as COCO, Kitti, and Open Images. We selected faster models trained on the COCO dataset to fine-tune them with the images from the Stanford car dataset. The fine-tuning is vital for our application because the pre-trained models trained on the COCO dataset; performance is not great as demonstrated in section.

4.3 Tensorflow Object Detection API for Fine-tuned Model

As stated earlier, the pre-trained models for object detection are not good at detecting the class required for our research. So, there is a need to fine-tune the pre-trained models with our dataset to be able to detect the car class 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 little time to train a model with transfer learning. We selected pre-trained models for object detection namely SSD MobileNET V1 with ResNet-101. The model is selected based on the literature review of object detection techniques describe in

Chapter 2. According to the discussion SSD MobileNET V1 model is the most accurate. The following section explains the fine-tuning of the pre-trained model.

4.3.1 Object Detection with SSD MobileNet

In this section, the pre-trained model of SSD MobileNet is fine-tuned with the training and validation datasets created in section 3.1. To begin we are using the provided configuration file as the basis of our fine-tuning. The configuration file requires a checkpoint file to initiate the fine tuning process which is already provided by TensorflowforSSDMobileNet. 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 shown in the following Figure 3.2, and ends with '.pbtxt' extension.

```
item {
  id: 1
  name: 'apple'
}
item {
  id: 2
  name: 'apricot'
}
item {
  id: 3
  name: 'avocado'
}
item {
  id: 4
  name: 'banana'
}
item {
  id: 5
  name: 'blackberries'
}
item {
  id: 6
```



```
name: 'blueberries'
}
item {
id: 7
name: 'cantaloupes'
}
item {
id: 8
name: 'cherries'
}
item {
id: 9
name: 'coconuts'
}
item {
id: 10
name: 'Guava'
}
item {
id: 11
name: 'grapes'
}
item {
id: 12
name: 'kiwi fruit'
}
item {
id: 13
name: 'lemons'
}
item {
id: 14
name: 'limes'
}
item {
id: 15
name: 'mangoes'
}
item {
id: 16
```

```
name: 'olives'
}
item {
id: 17
name: 'orange'
}
item {
id: 18
name: 'passion fruit'
}
item {
id: 19
name: 'pears'
}
item {
id: 20
name: 'pineapples'
}
item {
id: 21
name: 'pomegranates'
}
item {
id: 22
name: 'strawberries'
}
item {
id: 23
name: 'tomato'
}
item {
id: 24
name: 'watermelons'
}
item {
  id: 25
  name: 'bell pepper'
}
item {
  id: 26
```

```
    name: 'broccoli'
  }
  item {
    id: 27
    name: 'carrot'
  }
  item {
    id: 28
    name: 'cauliflower'
  }
  item {
    id: 29
    name: 'corn'
  }
  item {
    id: 30
    name: 'cucumber'
  }
  item {
    id: 31
    name: 'eggplant'
  }
  item {
    id: 32
    name: 'garlic'
  }
  item {
    id: 33
    name: 'ginger'
  }
  item {
    id: 34
    name: 'onion'
  }
  item {
    id: 35
    name: 'potato'
  }
  item {
    id: 36
```

```
    name: 'pumpkin'
  }
  item {
    id: 37
    name: 'white radish'
  }
  item {
    id: 38
    name: 'jackfruit'
  }
  item {
    id: 39
    name: 'lady finger'
  }
  item {
    id: 40
    name: 'bitter gourd'
  }
}
```

A sample of object label map file

The training record and test record file locations are added along with the object label map file as shown in below.

with the object label map file as shown in Figure 3.3 below.

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/object-detection.pbtxt"
}
eval_config: {
  num_examples: 1980
}
eval_input_reader: {
  tf_record_input_reader {
```

```

input_path: "data/test.record"
}
label_map_path: "training/object-detection.pbtxt"
shuffle: false
num_readers: 1
}

```

A sample of updated configuration file

```

item
  _scales: 0.25, 0.5, 1.0, and 2.0
  _ aspect ratios: 0.5, 1.0, and 2.0
  _ first stage maximum proposals: 300
  _ learning rate: initial value is 0.0003, reduce to 0.00003 after step 900,000
  and further reduce to 0.000003 at step 1,200,000
  _ data augmentation options: random horizontal flip
  _ batch size: 24

```

Development of overall mAP when fine-tuning SSD Mobilenet

The Faster RCNN model is fine-tuned for 100,000 steps as shown and the downward trend of total loss throughout the training process. The total loss values show a fluctuating trend as the batch size is 24, and a loss for each iteration can be slightly different from the previous iteration. However, the key point is that the loss values decreased overall during the course of training.

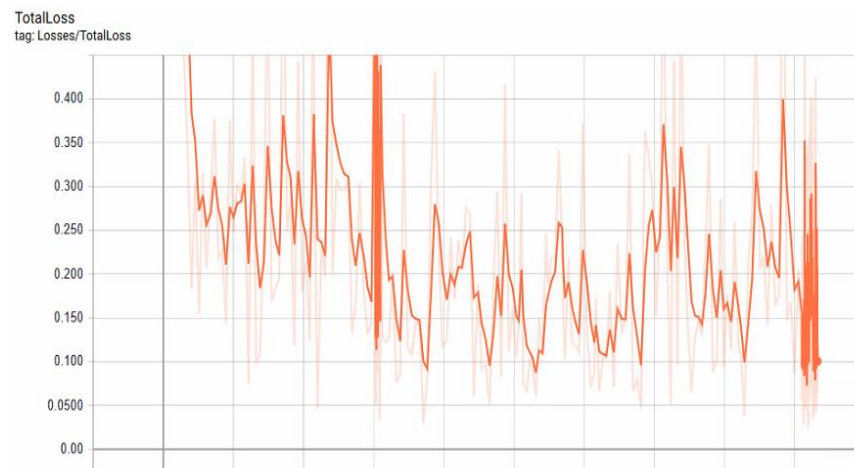


Figure 4.1: Decline of total loss when fine-tuning SSD MobileNet

4.3.2 Object Detection with Voice identification Android App For Visually Impaired Person

The SSD Mobile-net V1 model is the sole candidate in performing real time object detection in mobile that is it is the only suitable model for mobile hardware resources. After obtaining the trained model it is then it is applied in a android app. The app contains many classes among which Detector-Activity.java class is the main class, which interest classifier class . Here the main object detection occurs. The object titles are matched with labeled titles and sounds are played accordingly obtaining voice identification for the detected object. Google assistance which is already present in android mobiles from the lollipop version and up to the newer versions that will come is used so that a visual impaired person can access the android application by voice. As a result any android older version users that are bellow lollipop cannot use this application.

Chapter 5

Result

We have tried to calculate our model's accuracy by Intersection Over Union that is an evaluation matrix through which can calculate accuracy on a particular dataset. An output predicted box is generated by applying any kind of algorithms. In object detection the evaluation of IOU can be calculated with two things: 1. Ground Truth Boxes which we have got after labeling our sample images & 2. Predicted boxes from our specific model. Predicted boxes has been marked with red color and ground truth boxes with green color.



Figure 5.1 : Fruits and vegetables image with predicted boundary box

We calculate accuracy with iou. We consider 0.5 as a threshold as a good prediction. If IOU value is greater than equal 0.5 than it is a good prediction. After that we also calculate by consider threshold value as 0.6,0.7,0.8,0.9 . ifiou>=0.5 then this will go to TP.

For finding accuracy we use this formula

$$accuracy = \frac{TP}{all\ detection} \times 100$$

Then we find the accuracy from our model.

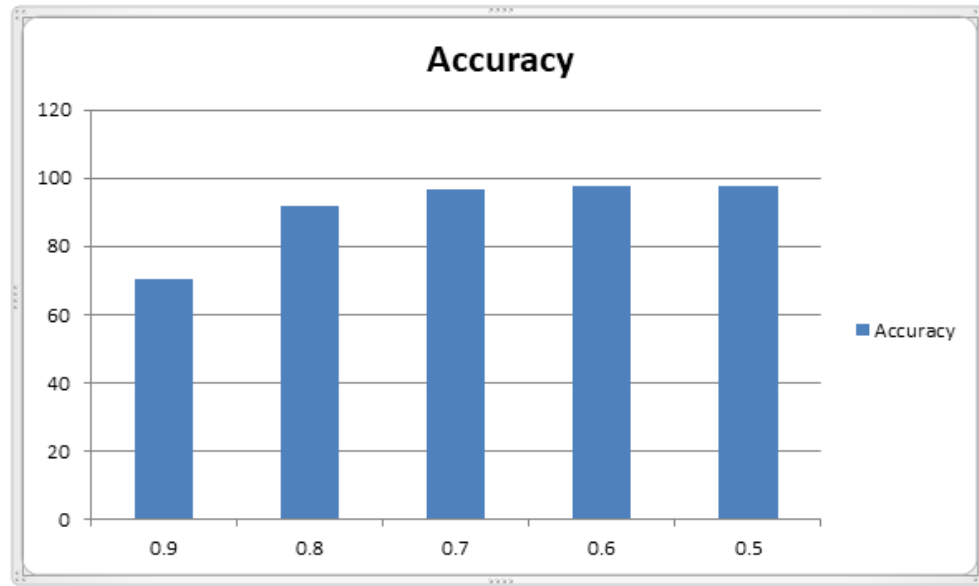


Figure 5.2:Accuracy graph

If we set threshold of IOU 0.5 then we get 89% accuracy. For the difference of IOU we get different accuracy.

Chapter 6

Conclusion

6.1 Summary

Fruits and vegetables detection and classification have great influence on the advances in the field of Intelligent Systems .Fruits and vegetables detection is a challenging task because there could be the variations of image light with different backgrounds and the application of it in android application is harder.We tried to use SSD Mobile-Net to detect fruits and vegetables. From our experiment we found 89% accuracy from our using model. Variance of light also make detection more challenging as well as the angle of images also creates more challenges to detect fruit and vegetables images. This model can only detect the rigid images of fruit and vegetables but performs worst foe the arbitrary angles of the image of this method is quite low so that machine or server does not wait a long to get the values. Machine does not remain idle for a long time.

6.2 Future Work

- However, there can be still some improvement to reduce the gap between training and test accuracy in our model. Other neural network models such as recurrent neural network [20], dilated convolutional neural network [21], etc can be applied to detect fruits and vegetables images. There are also some future works for the fruits and vegetables detection. We will try to build a better real time android app with more fruit and vegetable classes. Develop the user interface so it can be easier to use. Converting the application into more easily portable device for visual impaired people. Include a learn mode for the curious people who are eager to learn more about fruits and vegetables.

References:

- [2013 IJCV] [Selective Search] Selective Search for Object Recognition
- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.
- [2] M. A. Manzoor and Y. Morgan, “Vehicle Make and Model classification system using bag of SIFT features,” in Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual, pp. 1–5, IEEE, 2017.
- [3] M. C. Narhe and M. Nagmode, “Vehicle classification using SIFT,” International Journal of Engineering Research and Technology ESRSA Publications, 2014.
- [4] X. Ma and W. E. L. Grimson, “Edge-based rich representation for vehicle classification,” in Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, vol. 2, pp. 1185–1192, IEEE, 2005.
- [5] J.-W. Hsieh, L.-C. Chen, and D.-Y. Chen, “Symmetrical SURF and its applications to vehicle detection and vehicle make and model recognition,” IEEE Transactions on intelligent transportation systems, vol. 15, no. 1, pp. 6–20, 2014.
- [6] X. Li and X. Guo, “A HOG feature and SVM based method for forward vehicle detection with single camera,” in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on, vol. 1, pp. 263–266, IEEE, 2013.
- [7] S. Sivaraman and M. M. Trivedi, “Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis,” IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 4, pp. 1773–1795, 2013.
- [8] F. Han, Y. Shan, R. C. Sekander, H. S. Sawhney, and R. Kumar, “A two-stage approach to people and vehicle detection with HOG-based SVM,” in Performance Metrics for Intelligent Systems 2006 Workshop, pp. 133–140, 2006.

- [9] P. Negri, X. Clady, S. M. Hanif, and L. Prevost, “A cascade of boosted generative and discriminative classifiers for vehicle detection,” *EURASIP Journal on Advances in Signal Processing*, vol. 2008, p. 136, 2008
- [10] Y. Hua, J. Guo, and H. Zhao, “Deep belief networks and deep learning,” in *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, Jan 2015, pp. 1–4.
- [11] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, “Vehicle Type Classification Using Unsupervised Convolutional Neural Network,” in *2014 22nd International Conference on Pattern Recognition*, pp. 172–177, Aug 2014.
- [12] C.M.Bautista, C.A.Dy, M.I.Mañalac, R.A.Orbe, and M.Cordel, “Convolutional neural network for vehicle detection in low resolution traffic videos,” in *Region 10 Symposium (TENSYP)*, 2016 IEEE, pp. 277–281, IEEE, 2016.
- [13] H. Huttunen, F. S. Yancheshmeh, and K. Chen, “Car type recognition with deep neural networks,” in *Intelligent Vehicles Symposium (IV)*, 2016 IEEE, pp. 1115–1120, IEEE, 2016.
- [14] Y. Zhou, H. Nejati, T.-T. Do, N.-M. Cheung, and L. Cheah, “Image-based vehicle analysis using deep neural network: A systematic study,” in *Digital Signal Processing (DSP)*, 2016 IEEE International Conference on, pp. 276–280, IEEE, 2016.
- [15] J.E.Espinosa, S.A.Velastin, and J.W.Branch, “Vehicle Detection Using AlexNet and Faster RCNN Deep Learning Models: A Comparative Study,” in *International Visual Informatics Conference*, pp. 3–15, Springer, 2017.
- [16] S. Ren, K. He, R. Girshick, and J. Sun, “Faster RCNN: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, pp. 91–99, 2015.
- [17] Q. Fan, L. Brown, and J. Smith, “A closer look at Faster RCNN for vehicle detection,” in *Intelligent Vehicles Symposium (IV)*, 2016 IEEE, pp. 124–129, IEEE, 2016.
- [18] S.-C. Hsu, C.-L. Huang, and C.-H. Chuang, “Vehicle Detection using Simplified Fast RCNN,”
- [19] R. B. Girshick, “Fast RCNN,” *CoRR*, vol. abs/1504.08083, 2015.
- [20] M. Liang and X. Hu, “Recurrent convolutional neural network for object recognition,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition*

-
- (CVPR), June 2015, pp. 3367–3375.
 - [21] Y. Li, X. Zhang, and D. Chen, “Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes,” CoRR, vol. abs/1802.10062, 2018.
 - [Online]. Available: <http://arxiv.org/abs/1802.10062>
 - [22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587. IEEE, 2014.