Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

Real Time Object Detection using CNN based Single Shot Detector Model

Abhinav Juneja *

*Corresponding Author, Professor, Department of IT, KIET Group of Institutions, Delhi-NCR Ghaziabad, Uttar Pradesh, India. E-mail: abhinavjuneja1979@gmail.com

Sapna Juneja

Professor, Department of CSE, IITM Group of Institutions, Sonipat, Haryana, India. E-mail: sapnajuneja1983@gmail.com

Aparna Soneja

Student, Department of CSE, BMIET, Sonipat, Haryana, India. E-mail: aparnaaugust3@gmail.com

Saurav Jain

Student, Department of CSE, BMIET, Sonipat, Haryana, India. E-mail: souravjain540@gmail.com

Abstract

Object Detection has been one of the areas of interest of research community for over years and has made significant advances in its journey so far. There is a tremendous scope in the applications that would benefit with more innovations in the domain of object detection. Rapid growth in the field of machine learning has complemented the efforts in this area and in the recent times, research community has contributed a lot in real time object detection. In the current work, authors have implemented real time object detection and have made efforts to improve the accuracy of the detection mechanism. In the current research, we have used ssd_v2_inception_coco model as Single Shot Detection models deliver significantly better results. A dataset of more than 100 raw images is used for training and then xml files are generated using labellimg. Tensor flow records generated are passed through training pipelines using the proposed model. OpenCV captures real-time images and CNN performs convolution operations on images. The real time object detection delivers an accuracy of 92.7%, which is an improvement over some of the existing models already proposed earlier. Model detects hundreds of objects simultaneously. In the proposed model, accuracy of object detection significantly improvises over existing methodologies in practice. There is a substantial dataset to evaluate the accuracy of proposed model. The model may be readily useful for object detection applications including parking lots, human identification, and inventory management.

Keywords: Object Detection; Deep Learning; CNN; SSD; Tensor Flow; OpenCV.

DOI: 10.22059/jitm.2021.80025

Introduction

In the recent years, there has been an exponential progress in the field of machine learning and artificial intelligence which has led to improvement in accuracy, reduction in human efforts and failure rate. This development has played a commendable role in reducing processing time, which has further led to improvement in net productivity and corresponding reduction in the cost. To explore the application domain of machine learning systems, assume a situation of tracing our lost mobile in an untidy and messy house. It appears to be a cumbersome and frustrating task for anyone. It needs only a few milliseconds to track the Location of mobile. Well, this is precisely the power we can harness from these amazing object detection algorithms, which are at the bottom of heart the deep learning algorithms.

The current research work focuses on proposing an object detection model that can take input from the web camera, find location of the object through webcam, and classify object on screen for its appropriate category. Eventually, the goal of the current work on object detection is to take raw images as inputs, find location of that object in the given picture accurately and mask or classify object with appropriate categories.

Various Approaches to Object Detection Problem (Hernandez-Penaloza et al., 2017)

- ***Naïve way***

Divide the image into four parts i.e., upper left-hand side corner, upper right-hand side corner, lower left-hand side corner and lower right-hand side corner. Suppose we want to find the pedestrian in the image. Feed each of the four parts into a classifier. This will give output whether the image has pedestrian or not. If found, then mark that patch in the image.

- ***Increase number of divisions***

Increasing the number of patches or divisions solution is better than naïve approach. The only disadvantage here is that many bounding boxes are required about the same thing.

- ***Perform structural divison***

It overcomes the disadvantage of second approach. Divide image into 10*10 grid.Define the centroid of each patch. For each centroid , take different patches and pass them through classifier.

- ***To make it even more efficient***

Increase grid size and instead of three patches, take more patches. However, by increasing patches, it will be very difficult for classifier, so we should take selective patches instead of all.

- ***Using deep learning***

Deep learning is an expansion of the of machine learning concerned with algorithm inspired by function of brain (Zhang et al., 2020; Esteva et al., 2017; Bali et al., 2020). We use Deep Learning as our object detection approach because it gives the best performance out of all approaches. In deep learning, we do not take patches of the image sample rather complete image is forwarded to the neural network to reduce the dimensions. Neural network may be helpful to suggest selective patches and give predictions as close as to original bounding box as possible.

- ***Challenges for Object Detection using Machine Learning Algorithms***

Though classification of objects with human eye is an easy task, however for a machine, it is technologically challenging because human eye can do well in two-dimensional images also. There are some challenges, which object detection algorithms come across in real time.

- ***Localization***

It is difficult for an algorithm to find location of a single object inside image (Zhao et al., 2019; Ouadiay et al., 2018).

- ***Instance Segmentation***

After locating object in the given image, algorithm needs to segment or divide or separate that object from other objects (Hu et al., 2018).

- ***Classification of object in different categories***

Images that are taken as input should be of high quality and resolution., Illumination, viewpoint, size of object and its orientation also effects greatly (Klette, 2014; Arad et al., 2019).

- ***Occlusion***

It happens in the situation of two or more things coming too close to each other and either merging or combining altogether (Chandel & Vatta, 2015).

- ***Mirroring***

Object detection system must recognize mirror image of any object (Owen & Chang, 2019).

Object detection is a multi-disciplinary research area; it often involves fields of image processing, deep learning and computer vision. In our current experimental work, we have used the concepts of machine learning and computer vision for object detection in real time environment. The concepts used in our work include sliding windows, support vector machine, Principal component analysis (Mishra et al., 2017) and SSD (Single Shot Multibox Detector) (Redmon et al., 2016 and Phadnis et al., 2018). For the purpose of experimental work, TensorFlow with OpenCV is used. The real time object detection finds a lot of scope for applications which are now a days becoming necessity for the human processes. A few of the applications include facial recognition, industrial quality check, self-driving cars, optical character recognition, robotics, real time disease detection (Emami & Suciu, 2012; Alie et al., 2017; Memon et al., 2016; Hamad & Kaya, 2016; Aggarwal et al., 2019).

The remaining sections of the paper discuss the modelling of our object detection system. Section 2 discusses the related work and motivation for current work, Section 3 relates to uncovering the proposed approach and methodology, Section 4 discusses the results and outcomes of the experimental work. Section 5, summarizes the major achievements and outcomes of work and the potential future work that may inspire other researchers to explore further in the field of computer vision and AI.

Related Work and Motivation

Researchers have made a remarkable progress in object detection methods so far. There is a plethora of objects detection algorithms proposed by research community for detection of objects in real time. The current section explores various object detection algorithms, which motivated us to pursue the current experimental work.

CNN (Convolution Neural Network): CNN is a specialized category of the deep learning algorithm that may accept as an input some sample image and perform convolution operation to extract features from an input image and be able to differentiate each object from one other. (Alganci et al., 2020 and Rajaraman et al., 2019). The structural architecture of a CNN network (Fig. 1) is similar with the connectivity structure of human brain neurons. The initial layer of a CNN architecture is the Convolution Layer and the entity responsible for convolution is addressed as the kernel/filter (Zhang et al., 2017). This layer is responsible for mapping the meaningful features from an image. After the convolution function a piecewise activation function Rectified Linear also known as Relu is used, which produces non-linear transformation on the input. Next, Pooling (Scherer et al., 2010) is done to reduce size of convolved feature map. In order to reduce the computational power needed in processing the data through reduction of dimensions and further to only extract features that are dominant, pooling performs a very significant role. There are two dominant pooling variants. Max pooling (Christlein et al., 2020) is done if the convolved feature is of 4*4 and we need to have 2*2 blocks, then divide convolved feature map into 2*2 blocks and then find maximum element in each block. We can use the average pooling to find the average of each block. Finally, the last step is to classify the objects. CNN may only be able to predict the class of objects; it is not capable of telling the location of the objects. Therefore, it becomes necessary to select a large number of regions to overcome this (Tran et al., 2020; Sedghi et al., 2019). There are other filtering strategies (Shang, 2020), which can be helpful for feature identification and censoring, in this work the authors have devised a hybrid strategy for censoring.

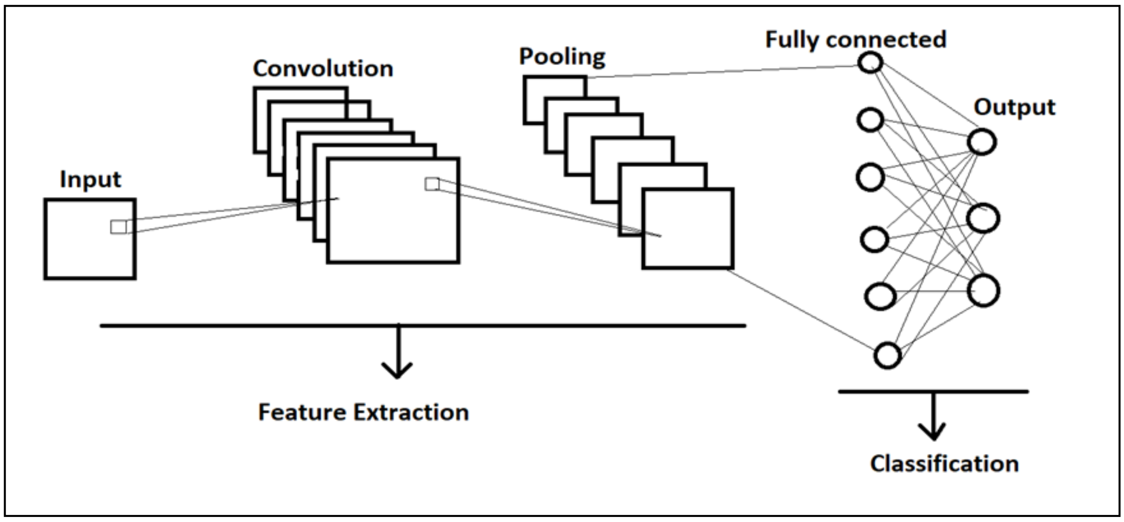


Figure 1. Classification Network using Convolutional layer

R-CNN: To tackle the limitation due to requirement for selecting a huge number of regions, researcher Ross Girshick illustrated a technique incorporating the selective search methods for extracting ‘2000’ regions of the image which are addressed as the region proposals (Ren, He, Girshick, & Sun, 2017). ‘2000’ regions are extracted from image and wrapped with a square, further they are transferred to the CNN, which serves as an extractor of the features. In order to evaluate the region proposal, Support Vector Machine (SVM) performs the classification of objects. It ingests a significant amount of time for training the network. The Selective search qualifies to be a fixed algorithm. There may be probabilities of creation of proposals of bad regions. (Lakhal et al., 2018)

Fast R-CNN: To make the R-CNN faster (Ren, He, Girshick, & Sun, 2017), improvised the procedure of training. In this model rather than feeding the proposal of regions, CNN receives the input image to generate a convolutional feature map and from that convolutional map, identification of region of proposals followed by wrapping into square. Regions of proposal of fixed size are shaped through ROI pooling(Qin et al., 2016) fixed size. The max pooling method invoked to convert the image features into corresponding region of image of dimension h*w into small static window of size H*W is addressed as ROI pooling. The region of Input is partitioned into H*W grids and then sub windows are created and finally maxpooling is applied to every grid.

Faster R-CNN: Faster R-CNN uses a detection pipeline concept to make the detection process even faster than R-CNN and Fast CNN(Ren, He, Girshick, & Sun, 2017).

RPN(Region Proposal Network) (Zhou et al., 2018) in Faster R-CNN: This method invokes the object proposals. RPN is composed of a classifier followed by a regressor. The function of classifier is to determine the probability of a candidate proposal possessing the object while the task of a regressor is to regress the coordinates corresponding to the proposals. Convolved featured map passes through a sliding window and central point of this sliding window is the anchor.

YOLO-You Only Look Once: YOLO (Redmon et al., 2016) is supplied an image as the input which is further split up into a grid and thereafter some bounding boxes are created inside the grid. Corresponding to every bounding box, network produces a probability of the class and the bounding box offset quantum values. A bounding box ensuring higher levels of probability designates. The method significantly processes faster compared to the faster R-CNN.

Mask R-CNN: Mask R-CNN (He et al., 2020) comprises of fixed level image segmentation and Faster R-CNN. Mask R-CNN made an improvisation over the ROI pooling layer and titled this as RoI Align layer. It fixes the location misalignment caused in ROI pooling layer.

SSD(Single Shot Multi-Box Detector): This method is popular for application in object detection (Liu et al., 2016) pertaining to the real time because of its accuracy and more speed (Fig. 2). Mean average of the precision (mAP) may be significant to determine the accuracy of the model. One disadvantage of YOLO was that it fails to detect object in the image if it is very small. While SSD can be used to detect small objects as well. SSD object detection consists of two components. The first one is VGG 16, which is responsible to extract the feature maps, and second one Conv4_3 layer (convolution) responsible for detecting the objects.

Every layer of SSD is able to detect and classify objects so accuracy increases. Leftmost layers or feature maps possessing higher level of resolutions are accountable for distinguishing small objects and rightmost layers can detect large objects. Each input image segments to 300*300*3. Then a series of multiple convolution layers, convolute the image.

Example: If we have feature map of 10*10 and convolution layer is of 1*1 with a stride of two, then output of conv layer will be 5*5. Stride refers to numb to the number of pixel shifts. We take conv4_3 to be 38*38. Corresponding to every cell, it furnishes '4' predictions of the objects. Each prediction comprises of the boundary box. Corresponding to each class there are 21 scores, with highest score selected for each class. The process of Generating of multiple predictions having boundary boxes and confidence becomes multi-box. SSD adds six convolutional layers after VGG16. These layers are 1*1 convolution with stride2. In those layers, we make six predictions. In total, SSD makes 8732 predictions. After discarding bounding boxes with low confidence, only top N predictions remain. This ensures removal of noisy predictions.

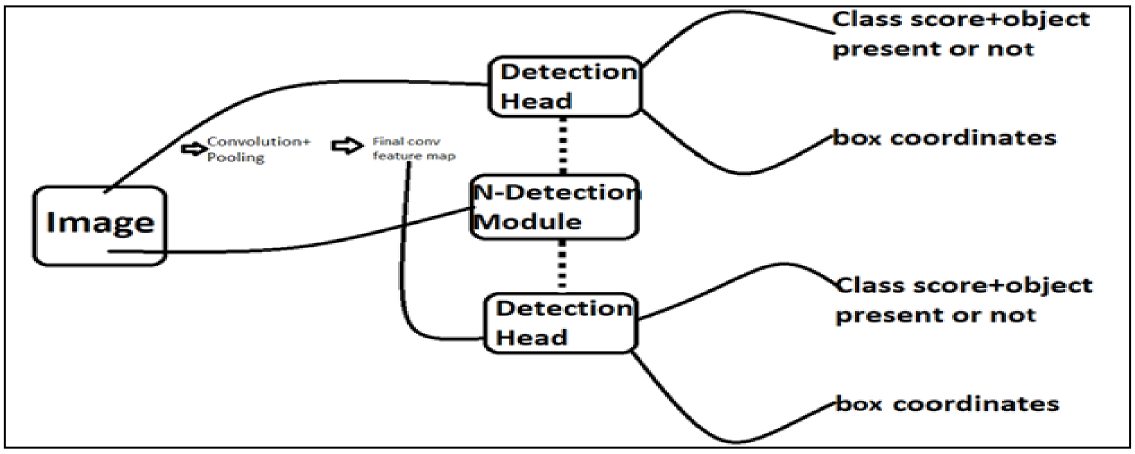


Figure 2. Block Diagram for Single Shot Multi-Box Detector (SSD)

MobileNet Vs Inception SSD

MobileNet Model (Phadnis et al., 2018 and Howard et al., 2017) is centered on convolutions which are depthwise separable, these are a class representing the factorized convolutions. The factorized convolutions convert a standard convolution to a corresponding depth wise convolution and further a 1*1 convolution, which is termed as pointwise convolution. In 'depth-wise' convolution, only one filter activates for every input channel. The Pointwise convolution invokes 1*1 convolution in order to combine the outcomes of depthwise convolution. Standard convolution filters and combines both inputs to a new group of outcomes in one-step. There is a splitting of 'depth-wise' separable convolution into two layers. The first layer is for filtering while another one for combining.

In the current experimental work, Inception SSD model is used. Inception SSD (Szegedy et al., 2016) is a complex network and is used to get better performance, both in terms of speed and prior to inception most of the CNN's just stacked convolution layers deeper to get better performance which were computationally expensive and were prone to overfitting. There are different versions of Inception model. The Inception1 (Szegedy et al., 2015) is a CNN that has a depth of 27 layers. The layered architecture of Inception1 possesses various Inception layers. Inception layer consists of mushroomed pool of numerous layers (a few of such layers include 1*1, 3*3 or 5*5 convolutional layer etc.), having the output filter bank of each of them coupled to a solo output vector materializing the feed for the next stage. Let us analyse what was the problem that inceptionv1 solved. Inceptionv2 posed to be a solution for the problem of excessive dimensional reduction seen by inception1 and further found out that factorization reduces complexity. This inception method factorizes

5*5 convolution to two 3*3 convolutions as 5*5 convolutions are expensive and slow. This changed improvised the performance of Inceptionv2. By invoking the segmentation of convolutions of the filter dimension n*n to an amalgamation of convolutions of dimensions 1*n and n*1. This was even further better. Here, instead of making deeper filter banks, wider banks come to use. We have used ssd_inception_v2_coco model as our pretrained model to configure our training pipeline. Table 1 shows a comparison of some of the relevant papers with the current work.

Table 1. Recent advancements in Object Detection Techniques

Author/s	Title	Objective and Outcome	Accuracy	Technique
Basri et al. (2018)	Faster R-CNN Implementation Method for Multi-Fruit Detection Using Tensorflow Platform	The model proposed method for detection of fruits. Method used Deep learning with the help of faster R-CNN for the classification of multiple fruits. Mango and papaya were inputs to the model. The work uses actual data of a farmer at the time of fruit harvest. The data is classified into two classes, first one being mango other being papaya.	70.6%	Faster R-CNN
Klette (2016)	Object Detection	The model proposes method for detection of the objects. The model implemented object detection on popular Pascal Voc dataset. The model was not reliable in occlusion.	72.4%	Single–shot MultiBox Architecture
Bashiri et al. (2018)	A fully-labelled image dataset to advance indoor objects detection	The work proposed a model for detection of the indoor objects. Model predicts on a fully labelled image dataset MCIndoor20000 on a fully trained neural network called Alexnet. The model used the concept of Transfer learning. Alexnet trains on 1.2 million images dataset.	64.4%	Alexnet
Howard et al. (2017)	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications	This object detection model uses the concept of MobileNets. MobileNets uses convolutions that are depthwise for building the light weighted deep neural networks. The authors present two unique global hyper-parameters that are able to convincingly balance among the latency and accuracy. In order to choose the appropriate sized model for an application these hyper-parameters are very helpful for the model builder. The MobileNets were effective in classification of objects, gains and even certain attributes of the face.	69.1%	CNN

Materials and Methods

Machine Learning Libraries Used

In the current research work, we have used some of the standard machine learning libraries for the purpose of performing object detection. There are two libraries which have been used here, these are TensorFlow and OpenCV. The various inherent features of these libraries which made us to use them are described below.

TensorFlow (Mulfari et al., 2017): Tensor Flow is one of the most famous open source deep learning library used across the globe by machine learning enthusiasts. The library implements its functionality in three sequential steps. Initially starting with the Preprocessing of data. It is that step in which data gets transformed or encoded to bring it to such a state that now the machine can parse it. It includes handling categorical values,dropping multicollinearity etc. Categorical variables are the variables that are discrete and not continous.Ordinal and nominal are two types of categorical variables. Ordinal variable can be ordered example t-shirt sizes. Nominal can’t be ordered. For instance, for color we cannot say blue is less than green or vice versa. Nominal and ordinal categorical variables should be processed differently.Then next we drop the Multicollinearity. It occurs in dataset when we have features which are strongly dependent on each other. For instance, Weight and Blood pressure have strong correlation. After preprocessing of data, we build the model, train it and test it.

OpenCV(Khan et al., 2019): This python library has specialization of solving the computer vision issues. The domain of Computer Vision pertains to furnish the methods to aid the computers visualize and comprehend the features of digital images including the photos and video. OpenCV with python uses the Numpy, that is a specialized library for performing the numerical calculations. All arrays structures of OpenCV are transformed into the Numpy arrays. In order to see, analyse and extract information from images OpenCV is instrumental(Guennouni et al., 2015). We can find an object in an image using OpenCV’s cv2.matchTemplate() function. Using OpenCV, we can load input image and convert it into gray, create bounding boxes across the object and remove noise from the image.

Process flow of the Proposed model

As the initial step of the experimental process, raw images were collected to make the dataset with the help of internet (Bashiri et al., 2018). We have made a huge collection of similar images and grouped them. Figure 3 shows the steps the sequential process followed in modelling the current system. It was ensured that images were taken from multiple angles, also care was taken about the brightness, scale, conditions of lightning and angles. The format of all the images used is .jpg and overall approximately 150 images of each category were collected, which gave satisfactory performance in detection at the end. All the collected images were properly stored in the directory made for storage. Out of the total collection, images were split into two sections, one section for training and the other for testing. A ratio of 90:10 was taken for in this work, it may even be taken to a factor of 80:20.

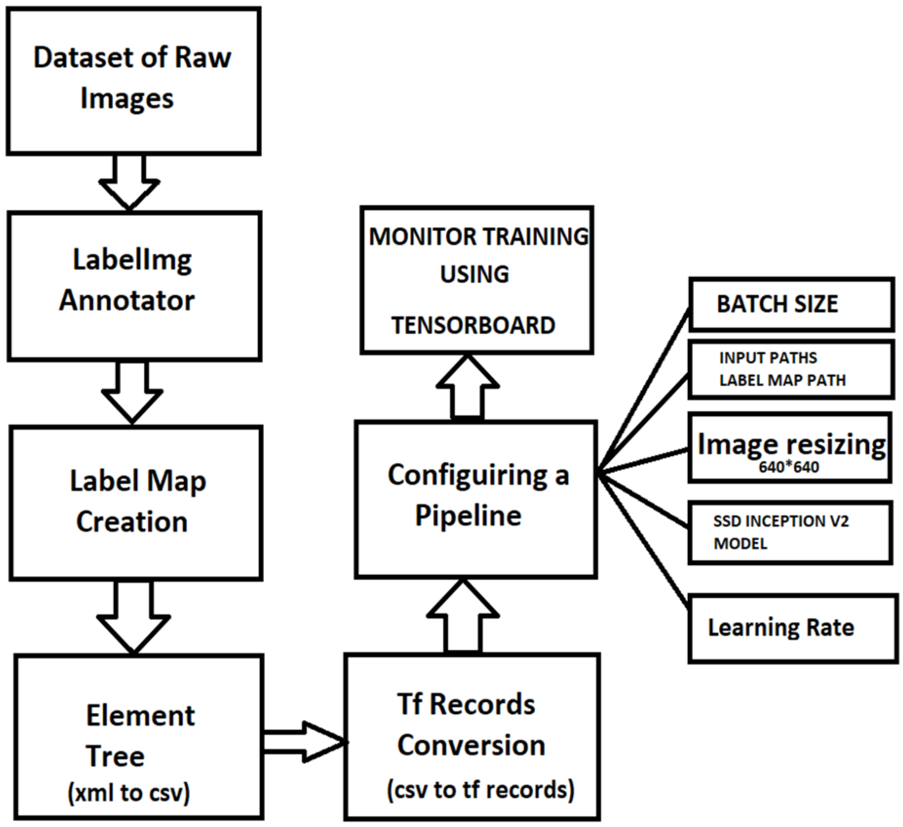
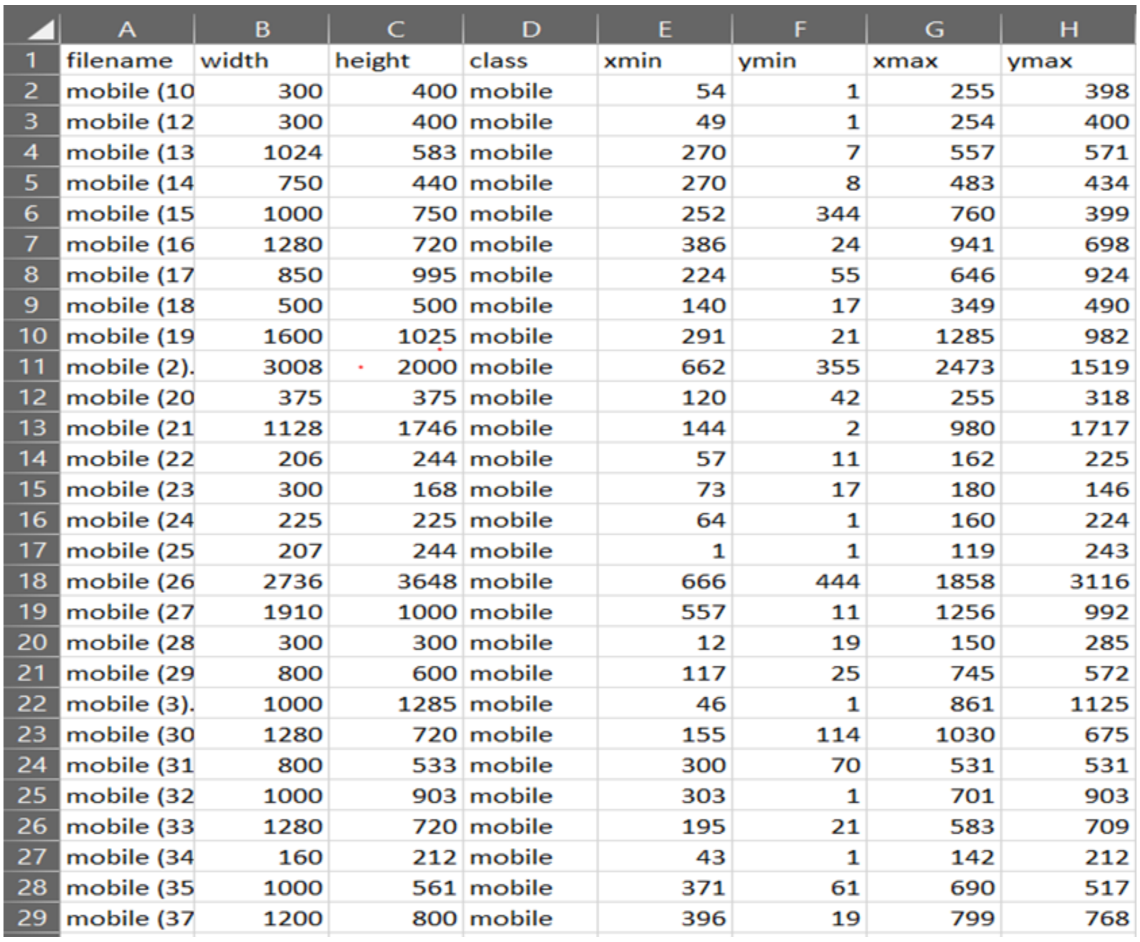


Figure 3. Object Detection Process for proposed model

During the next phase, images were properly labelled manually using a labelling image annotator called LabelImg. This annotator provides a user- friendly GUI and its saves label files in PascalVOC format which can be useful later on(Fiedler et al., 2019). Once the process of annotation of the image dataset is completed, as a common practice only a portion of it is used for training, and the remaining is kept for the evaluation purposes. Next we create the xml files for each of the labeled image. XML is an intrinsically ordered data format and best method of representation for XML is through a tree. ElementTree(Tu et al., 2004) denotes the complete XML document in the form of a tree and element is represented by a single node of the tree. Any communication with the document is made at ET level and within XML elements it is done on Element level. Further a conversion of XML to cs is done for all files due to the fact that general users are not able to read the data in XML format(Mitlohner et al., 2016). CSV format separates values using commas or delimeters. XML files can be passed using python built-in library called ElementTree. Figure 4 shows the snapshot of csv file created for the images. Further in the next phase we converted the csv files to TensorFlow Records(Smith et al., 2016), also addressed as TF records. In this approach whatever data is available is coverted into the supported format.



	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	mobile (10	300	400	mobile	54	1	255	398
3	mobile (12	300	400	mobile	49	1	254	400
4	mobile (13	1024	583	mobile	270	7	557	571
5	mobile (14	750	440	mobile	270	8	483	434
6	mobile (15	1000	750	mobile	252	344	760	399
7	mobile (16	1280	720	mobile	386	24	941	698
8	mobile (17	850	995	mobile	224	55	646	924
9	mobile (18	500	500	mobile	140	17	349	490
10	mobile (19	1600	1025	mobile	291	21	1285	982
11	mobile (2).	3008	2000	mobile	662	355	2473	1519
12	mobile (20	375	375	mobile	120	42	255	318
13	mobile (21	1128	1746	mobile	144	2	980	1717
14	mobile (22	206	244	mobile	57	11	162	225
15	mobile (23	300	168	mobile	73	17	180	146
16	mobile (24	225	225	mobile	64	1	160	224
17	mobile (25	207	244	mobile	1	1	119	243
18	mobile (26	2736	3648	mobile	666	444	1858	3116
19	mobile (27	1910	1000	mobile	557	11	1256	992
20	mobile (28	300	300	mobile	12	19	150	285
21	mobile (29	800	600	mobile	117	25	745	572
22	mobile (3).	1000	1285	mobile	46	1	861	1125
23	mobile (30	1280	720	mobile	155	114	1030	675
24	mobile (31	800	533	mobile	300	70	531	531
25	mobile (32	1000	903	mobile	303	1	701	903
26	mobile (33	1280	720	mobile	195	21	583	709
27	mobile (34	160	212	mobile	43	1	142	212
28	mobile (35	1000	561	mobile	371	61	690	517
29	mobile (37	1200	800	mobile	396	19	799	768

Figure 4. Test_labels.csv file showing parameters of images

After this the next step was to configure the training pipeline(Sugimura & Hartl, 2018). The model used in this project is ssd_inception_v2_coco model which has been discussed in previous sections. Training efforts for training the ssd_inception_v2 (our chosen model) is dependent on various factors incluidng Computational power of hardware, whether TensorFlow GPU or CPU is used, how big dataset size is and complexity of objects taken. We trained our model for around 20 days on a normal pc using TensorFlowGPU. TensorBoard (Dignam et al., 1983) shows our training progress in the form of a graph. Through tensor board it is very convenient to visualize loss and accuracy, understand the quality aspects of model training, Viewing histograms , graphs etc. and displaying images text etc.

Results

The object detection in real time using the modern machine learning tools have made several applications a new way of looking at the world. With all the background knowledge and tools to support our experimental work, an SSD Model interacts with our data. Finally, when we run our model, it opens a separate window that is able to detect multiple objects simultaneously. In Fig.5, we can see it detects mobile phone and remote simultaneously using a running webcam. We ran tests with dataset built for around 100 objects such as mobile, bag, remote, book, chair, table etc. and about 100-150 images for each object that means for total database of about 10000 images, out of which about 9000 images trained the model and remaining 1000 tested it. We obtained overall success rate of about 92.4%. One of the best cases include that of a cell phone with over 97% accuracy. The video window is updates by the program through a new frame in a regular interval of 0.25 and 0.5 second, this implies that an average of 2 - 4 FPS.As we have made the model using SSD, so its speed and overall accuracy is better than other models. The time to identify an object depends more or less linearly on the number of key features fed to the system, and size of the database. Presently, the overall recognition time on a single processor is about 20 seconds for the 6-object database, and 2 min for the 24-object database.

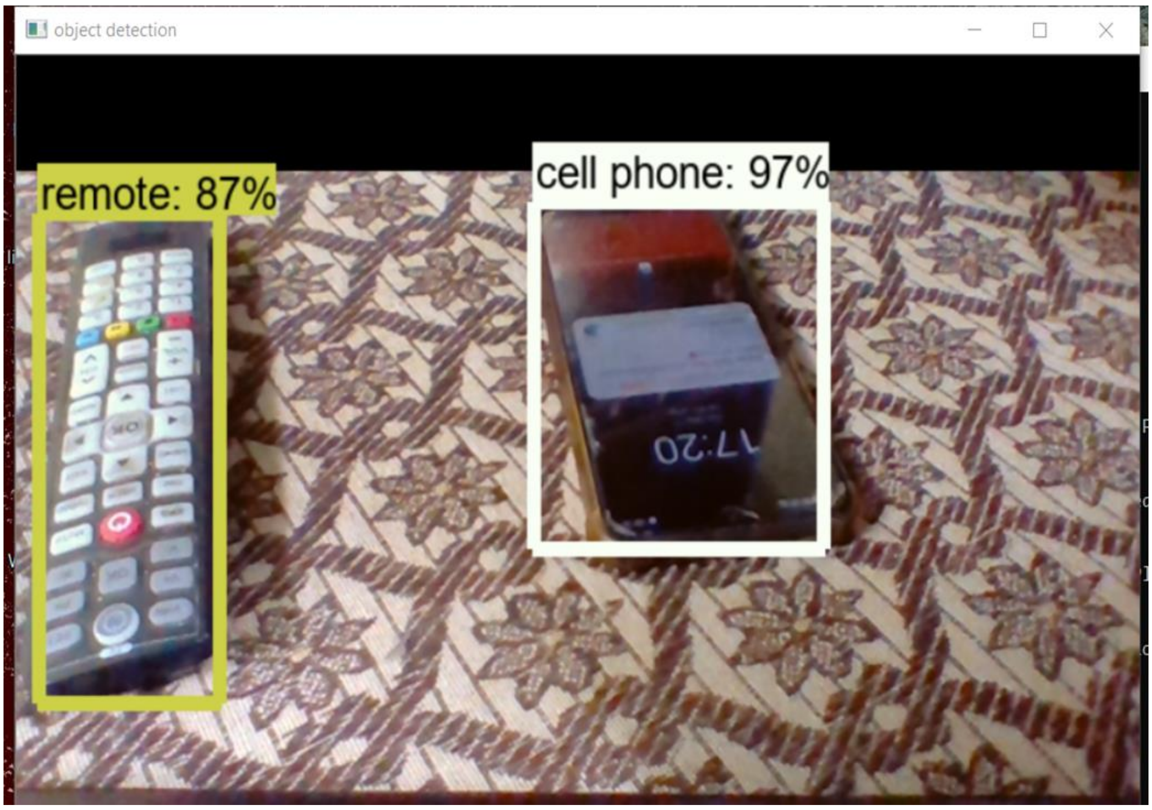


Figure 5. Algorithm successfully detecting objects

There are many papers published on object detection with various technologies in it having various accuracies associated with it. By using our methodology, we are able to get a better accuracy (mAP values) than other models Table 2 shows a comparative analysis of relevant work.

Table 2. Analysis of outcomes of related work to the current work

Author/s	Title	Outcome Average	Technique
Basri et al. (2019)	Faster R-CNN Implementation Method for Multi-Fruit Detection Using Tensorflow Platform	Accuracy – 70.6%	Faster R-CNN
Klette (2014)	Object Detection	Accuracy- 72.4%	Single – shot MultiBox Architecture
Bashiri et al.(2018)	A fully-labelled image dataset to advance indoor objects detection	Accuracy-64.4%	Alexnet
Howard et al. (2017)	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications	Accuracy-69.1%	CNN
Zhou et al. (2018)	Object Detection from Images Based on MFF-RPN and Multi-scale CNN	Accuracy- 72.8%	MFF-RPN and Multi-scale CNN
Chandel & Vatta (2015)	Occlusion Detection and Handling: A Review	Accuracy – 91.3%	Occlusion
Current work	Object Detection Using CNN based Single Shot Detector Model	Accuracy- 92.4%	Using CNN based Single Shot Detector Model

Discussion

From the current research work, we are able to say that we can make a real-time object detection program using TensorFlow GPU and OpenCV, which is able to capture the Images using webcam and segment them, then classify them as per their categories. This paper provides a detailed review of different approaches for object detection and how can one use deep learning for getting best results. We got inspired for this research from different models used in object detection including methods like CNN, R CNN, Faster RCNN and SSD, therefore this paper discusses all of these. Paper explains the need to use SSD for our model. Current work substantially deviates from the methodology used by Phadnis (Phadnis et al., 2018) as they used Mobile net SSD whereas we used Inception SSD. For developing this model, we collected raw images of different angles and brightness from the internet, labelled them and partitioned them into training and testing, converted them into xml files using an image tagger and then into CSV and TensorFlow records. We got a good result from self-made dataset and pretrained ssd_inceptionv2_coco model. As it is clear from the table II above that we were able to detect objects on a running webcam with about 92.4% accuracy.

Conclusion

Currently, there are many object detection models for detecting the objects. During the current experimental effort, SSD has been used. Deciding on which particular model is the best is a very typical and complex task as that some models may possess better accuracy but may not deliver the best processing speed and vice-versa.

In case when the objects are large, SSD can outperform Faster R-CNN in accuracy. SSD is fast but SSD can exhibit lower performance than Faster R-CNN in case of small objects. SSD paired with the MobileNet provides best accuracy. mAP has been evaluated with PASCAL VOC 2012 testing set. We can see that SSD@512, YOLO perform considerably good in terms of accuracy. In terms of frames per second also, SSD and YOLO perform better than other models. Fig 6, here exhibits the comparative performance of various object detection models for key attributes.(Sanjay & Ahmadinia, 2019).

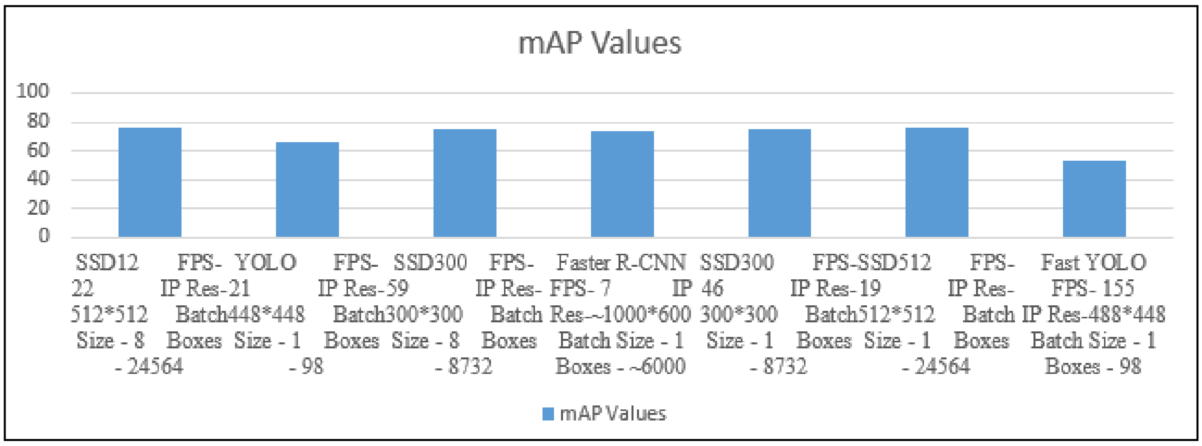


Figure 6. Relative performance exploration of various detection models

Resolution of the input image has a significant impact on the accuracy of the model. Fig 7. Shows the GPU time for different models, which depicts that SSD performs best among all as it takes least GPU time as shown in the graph. In terms of memory, SSD with MobileNet is best because of its minimal memory requirements. Fast R- CNN attained a mAP of 66.9, while expending networks trained on PASCAL VOC 2007 training data. Faster R-CNN delivered a moderately better performance with a mAP of 69.9. SSD attained a mAP of 68.0 corresponding to an input dimension of 300*300 and 71.6 corresponding to the input dimension 512*512. In the current work, we have done labelling of images by labelimg and converted them into xml file and further converted xml to tf records and then finally performed training on the ssd model. This methodology has been quite useful in increasing the efficiency and accuracy of the model while compared to the other technologies. Inception SSD is beneficial for object detection modelling both in terms of speed and prior to inception most of the CNN's just stacked convolution layers deeper to get better performance which were computationally expensive and were prone to overfitting.

Object detection is a significant technology in today's AI systems and its future scope includes mainly driverless cars, facial recognition, robotics, industrial goods check-up and many more. The current work may help the researchers to understand the concepts of object detection in the modern world in a more explicit manner. There are still many open areas including detection with accuracy in varying conditions on real time environment, which will further gain more improvement in coming years. Our proposed method may increase the efficiency of the modern day applications including functional requirements to count number of peoples in a moving pedestrian subway, detection of vehicles in an area, for industrials checks and it can be used for many traffic violations like red light etc.

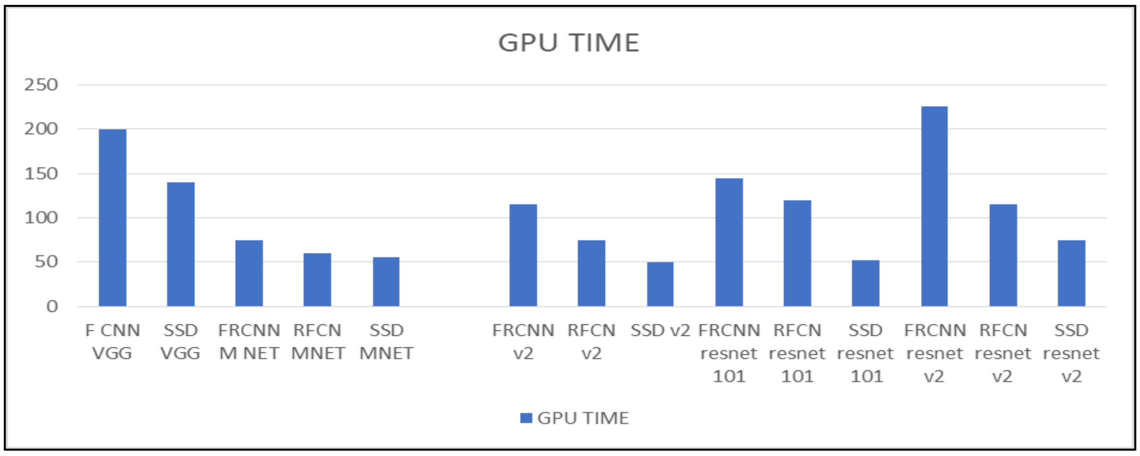


Figure 7. GPU time for different object detection models

References

Aggarwal, D., Bali, V., & Mittal, S. (2019). An insight into machine learning techniques for predictive analysis and feature selection. *International Journal of Innovative Technology and Exploring Engineering*, *8 Special*(9), 342–349. https://doi.org/10.35940/ijitee.I1055.0789S19

Alganci, U., Soydas, M., & Sertel, E. (2020). Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. *Remote Sensing*, *12*(3). https://doi.org/10.3390/rs12030458

Alie, N. M., Karis, M. S., Wong, G. J., Bahar, M. B., Sulaiman, M., Ibrahim, M. M., & Abidin, A. F. Z. (2017). Quality checking and inspection based on machine vision technique to determine tolerancevalue using single ceramic cup. *ARPN Journal of Engineering and Applied Sciences*, *12*(8), 2737–2742.

Arad, B., Kurtser, P., Barnea, E., Harel, B., Edan, Y., & Ben-Shahar, O. (2019). Controlled lighting and illumination-independent target detection for real-time cost-efficient applications. The case study of sweet pepper robotic harvesting. *Sensors (Switzerland)*, *19*(6), 1–15. https://doi.org/10.3390/s19061390

Bali, V., Kumar, A., & Gangwar, S. (2020). A novel approach for wind speed forecasting using LSTM-ARIMA deep learning models. *International Journal of Agricultural and Environmental Information Systems*, *11*(3), 13–30. https://doi.org/10.4018/IJAEIS.2020070102

Bashiri, F. S., LaRose, E., Peissig, P., & Tafti, A. P. (2018). MCIndoor20000: A fully-labeled image dataset to advance indoor objects detection. *Data in Brief*, *17*, 71–75. https://doi.org/10.1016/j.dib.2017.12.047

Basri, H., Syarif, I., & Sukaridhoto, S. (2019). Faster R-CNN implementation method for multi-fruit detection using tensorflow platform. *International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2018 - Proceedings*, 337–340. https://doi.org/10.1109/KCIC.2018.8628566

Chandel, H., & Vatta, S. (2015). Occlusion Detection and Handling: A Review. *International Journal of Computer Applications*, *120*(10), 33–38. https://doi.org/10.5120/21264-3857

Christlein, V., Spranger, L., Seuret, M., Nicolaou, A., Kr, P., & Maier, A. (2020). *Deep Generalized Max Pooling*. *2020*(211).

Dignam, J. D., Martin, P. L., Shastry, B. S., & Roeder, R. G. (1983). Eukaryotic gene transcription with purified components. *Methods in Enzymology*, *101*(C), 582–598. https://doi.org/10.1016/0076-6879(83)01039-3

Emami, S., & Suciu, V. P. (2012). Facial Recognition using OpenCV. *Journal of Mobile, Embedded and Distributed Systems*, *4*(1), 38–43. http://www.jmeds.eu/index.php/jmeds/article/view/57

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

Fiedler, N., Bestmann, M., & Hendrich, N. (2019). ImageTagger: An Open Source Online Platform for Collaborative Image Labeling. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11374 LNAI*(January), 162–169. https://doi.org/10.1007/978-3-030-27544-0_13

Guennouni, S., Ahaitouf, A., & Mansouri, A. (2015). Multiple object detection using OpenCV on an embedded platform. *Colloquium in Information Science and Technology, CIST*, *2015-Janua*(January), 374–377. https://doi.org/10.1109/CIST.2014.7016649

Hamad, K., & Kaya, M. (2016). A Detailed Analysis of Optical Character Recognition Technology. *International Journal of Applied Mathematics, Electronics and Computers*, *4*(Special Issue-1), 244–244. https://doi.org/10.18100/ijamec.270374

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2020). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 386–397. https://doi.org/10.1109/TPAMI.2018.2844175

Hernandez-Penaloza, G., Belmonte-Hernandez, A., Quintana, M., & Alvarez, F. (2017). A Multi-Sensor Fusion Scheme to Increase Life Autonomy of Elderly People with Cognitive Problems. *IEEE Access*, *6*(c), 12775–12789. https://doi.org/10.1109/ACCESS.2017.2735809

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. http://arxiv.org/abs/1704.04861

Hu, R., Dollar, P., He, K., Darrell, T., & Girshick, R. (2018). Learning to Segment Every Thing. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4233–4241. https://doi.org/10.1109/CVPR.2018.00445

Khan, M., Chakraborty, S., Astya, R., & Khepra, S. (2019). Face Detection and Recognition Using OpenCV. *Proceedings - 2019 International Conference on Computing, Communication, and*

Intelligent Systems, ICCCIS 2019, *2019-Janua*, 116–119. https://doi.org/10.1109/ICCCIS48478.2019.8974493

Klette, R. (2014). *Object Detection*. *163050048*, 375–413. https://doi.org/10.1007/978-1-4471-6320-6_10

Lakhal, M. I., Çevikalp, H., Escalera, S., & Ofli, F. (2018). Recurrent neural networks for remote sensing image classification. *IET Computer Vision*, *12*(7), 1040–1045. https://doi.org/10.1049/iet-cvi.2017.0420

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Memon, Q., Ahmed, M., Ali, S., Memon, A. R., & Shah, W. (2016). Self-driving and driver relaxing vehicle. *2016 2nd International Conference on Robotics and Artificial Intelligence, ICRAI 2016*, *November 2016*, 170–174. https://doi.org/10.1109/ICRAI.2016.7791248

Mishra, S., Sarkar, U., Taraphder, S., Datta, S., Swain, D., Saikhom, R., Panda, S., & Laishram, M. (2017). Multivariate Statistical Data Analysis- Principal Component Analysis (PCA). *International Journal of Livestock Research*, *January*, 1. https://doi.org/10.5455/ijlr.20170415115235

Mitlohner, J., Neumaier, S., Umbrich, J., & Polleres, A. (2016). Characteristics of open data CSV files. *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*, *838*, 72–79. https://doi.org/10.1109/OBD.2016.18

Mulfari, D., Longo Minnolo, A., & Puliafito, A. (2017). Building tensor flow applications in smart city scenarios. *2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017*. https://doi.org/10.1109/SMARTCOMP.2017.7946991

Ouadiay, F. Z., Bouftaih, H., Bouyakhf, E. H., & Himmi, M. M. (2018). Simultaneous object detection and localization using convolutional neural networks. *2018 International Conference on Intelligent Systems and Computer Vision, ISCV 2018*, *2018-May*, 1–8. https://doi.org/10.1109/ISACV.2018.8354045

Owen, D., & Chang, P.-L. (2019). *Detecting Reflections by Combining Semantic and Instance Segmentation*. 1–12. http://arxiv.org/abs/1904.13273

Phadnis, R., Mishra, J., & Bendale, S. (2018). Objects Talk - Object Detection and Pattern Tracking Using TensorFlow. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, 1216–1219. https://doi.org/10.1109/ICICCT.2018.8473331

Qin, Y., He, S., Zhao, Y., & Gong, Y. (2016). *RoI Pooling Based Fast Multi-Domain Convolutional Neural Networks for Visual Tracking*. *133*, 198–202. https://doi.org/10.2991/aiie-16.2016.46

Rajaraman, S., Sornapudi, S., Kohli, M., & Antani, S. (2019). Assessment of an ensemble of machine learning models toward abnormality detection in chest radiographs. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 3689–3692. https://doi.org/10.1109/EMBC.2019.8856715

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Ren, S., He, K., Girshick, R., Zhang, X., & Sun, J. (2017). Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(7), 1476–1481. https://doi.org/10.1109/TPAMI.2016.2601099

Sanjay, N. S., & Ahmadinia, A. (2019). MobileNet-Tiny: A deep neural network-based real-time object detection for rasberry Pi. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, 647–652. https://doi.org/10.1109/ICMLA.2019.00118

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6354 LNCS*(PART 3), 92–101. https://doi.org/10.1007/978-3-642-15825-4_10

Sedghi, H., Gupta, V., & Long, P. M. (2019). *The Singular Values of Convolutional Layers*. 1–12.

Shang, Y. (2020). Consensus of Hybrid Multi-Agent Systems with Malicious Nodes. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *67*(4), 685–689. https://doi.org/10.1109/TCSII.2019.2918752

Smith, R., Gu, C., Lee, D. S., Hu, H., Unnikrishnan, R., Ibarz, J., Arnoud, S., & Lin, S. (2016). End-to-end interpretation of the French street name signs dataset. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9913 LNCS*(June), 411–426. https://doi.org/10.1007/978-3-319-46604-0_30

Sugimura, P., & Hartl, F. (2018). *Building a Reproducible Machine Learning Pipeline*. http://arxiv.org/abs/1810.04570

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Tran, D. S., Ho, N. H., Yang, H. J., Baek, E. T., Kim, S. H., & Lee, G. (2020). Real-time hand gesture spotting and recognition using RGB-D Camera and 3D convolutional neural network. *Applied Sciences (Switzerland)*, *10*(2). https://doi.org/10.3390/app10020722

Tu, T., O'Hallaron, D. R., & López, J. C. (2004). Etree: A database-oriented method for generating large octree meshes. *Engineering with Computers*, *20*(2), 117–128. https://doi.org/10.1007/s00366-004-0283-5

Zhang, Y., Kong, J., Qi, M., Liu, Y., Wang, J., & Lu, Y. (2020). Object detection based on multiple information fusion net. *Applied Sciences (Switzerland), 10*(1). https://doi.org/10.3390/app10010418

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

Zhou, J., Zheng, H., Yin, H., & Chai, Y. (2018). Object detection from images based on MFF-RPN and multi-scale CNN. *Lecture Notes in Electrical Engineering*, *460*, 343–351. https://doi.org/10.1007/978-981-10-6499-9

Bibliographic information of this paper for citing:

Juneja, Abhinav; Juneja, Sapna; Aparna & Jain, Saurav (2021). Real Time Object Detection using CNN based Single Shot Detector Model. *Journal of Information Technology Management*, 13(1), 62-80.