# Automated and Curated Sack Count Leveraging Video Analysis on Moving Objects

**Ritin Behl, Harsh Khatter, Prabhat Singh, Garima Bhardwaj, and Prateek Chaturvedi**

## 1 Introduction

From the advances in car technology to the various features that help drivers in modern cars, the long way of automotive technology is becoming increasingly sophisticated with the Laptop vision that results in greater and easier. At the same time, infrastructure transportation providers and resources will increase their reliance on CV to improve their safety and efficiency in transportation. The concept of computer in this way helps to solve critical problems at both the transit level, the consumer-level, and the infrastructure providers [1].

The growing demand for limited transportation infrastructure is constantly growing and leading to traffic delays, accidents, and traffic congestion have also had serious economic consequences. Technological advances such as the concept of computers play a key role in solving these types of problems in efficient and effective ways such as traffic charging, incident detection and management, traffic

R. Behl
Department of Information Technology, ABES Engineering College, Ghaziabad, India

ABES Engineering College, Ghaziabad, India
e-mail: ritin.behl@abes.ac.in

H. Khatter · P. Singh
ABES Engineering College, Ghaziabad, India

Department of Computer Science and Engineering, ABES Engineering College, Ghaziabad, India
e-mail: harsh.khatter@abes.ac.in; prabhat.singh@abes.ac.in

G. Bhardwaj · P. Chaturvedi (✉)
AUGN, ABES Engineering College, Ghaziabad, India

Amity University Greater Noida Campus, Greater Noida, Uttar Pradesh, India
e-mail: gbhardwaj@gn.amity.edu; pchaturvedi1@gn.amity.edu

S. Awasthi et al. (eds.), *Artificial Intelligence for a Sustainable Industry 4.0*,
https://doi.org/10.1007/978-3-030-77070-9_17

monitoring, traffic monitoring and vehicle control and many more alternatives; and a wide range of applications that computer vision technology can support. Self-help assistance programs are being rolled out at increasing prices, these programs will begin to shift their role from one aid to another to facilitate decisions as they are related to safety but also the power that goes up to connect the infrastructure [2].

The aim of the independent acquisition programs is to reduce traffic congestion, to produce less expensive cars and emergency equipment, to improve public safety, to reduce environmental impacts, to improve mobility data, and so on. The technology helps regions, cities, and towns at the national level to meet the growing demands of a better travel system. The effectiveness of an independent system is largely based on the efficiency and completeness of the acquisition technology [3, 4].

Car detection by video cameras is one of the most attractive and inaccessible technologies that require large data collection and use of vehicle control and management systems. Object discovery is also the basis for tracking. Proper discovery of an object results in better tracking of the object. Modern computer-controlled systems have more complex access requirements than those used by conventional traffic controllers as one of the road signs, for which many explosive devices are made [5].

## 2 Problem Statement

Often it happens, human observation is required to keep track of objects unloaded from the truck to the warehouse or loaded from the warehouse into the trucks. There is a possibility of errors. To solve this problem a web application will be designed which will count the number of sacks that are loaded into the truck or unloaded from the truck. A camera would be used which will be monitoring the sacks that are loaded and unloaded, and it would be easier to find if the sacks loaded are equal to the number of sacks unloaded or not. With the advancement in video analytics, it is feasible to remove the human observation and replace it with a camera-based video analytics solution by object detection method [6–8].

This sack counting system is one of the smart solutions for the transportation industry. There is a chance of error and false results in human observation. The goal of the sack counting system backend engine is to process the image frames passed to it via the image processing algorithm according to architectural diagram. The backend engine is trained as such, that, it processes the image frames which was found in the frame folder and predicts the sack count using predictive algorithm and returns live counter of the sacks loaded on to the truck. The engine used based on Tensorflow v2 frame work, and the image frames are fetched from the live video recording from the CCTV camera recording the video [9, 10].

The development of the project will flow in steps one after another such as selecting a model, adapting an existing dataset that creates and annotate own dataset, modifying the model configure the file, trains the model and saving the model, moves the model in another piece of software, and finally displays the result of backend engine on the custom-developed User Interface in the form of web application.

The data would be collected from live camera feed, and then the collected video would be converted into frames. The frames will be processed then, to extract the required information, i.e., the sacks. The model building algorithm would learn from the frames about the color, shape, size, and position of the sacks. Then the model will be tested against some data which is not used for training.

To count the number of sacks a region of interest (ROI) will be determined. The sacks crossing beyond the ROI would be counted and the counter will be incremented by 1. Initially, the counter would be set to zero. The counter will be refreshed every time the model is run again after terminating the model once. The result output is the command prompt screen of the sack count system containing the number of sacks either loaded or unloaded from the truck. Also the annotated frames with confidence score is also displayed on the output window [11].

## *2.1 Related Work*

Object acquisition is a common term used for CV techniques that classify and locate objects in a photograph. The discovery of the modern object depends on the use of the CNN network. There are many programs suitable for Faster R-CNN, R-FCN, Multibox single Shot Detector (SSD), and YOLO (You Just Look Once). The original R-CNN methods work with neural net classifier performance on plant sample images using external suggestions (samples are planted with external box suggestions, with the extraction feature performed on all planted samples). This method is very expensive due to the many crops, the fast R-CNN which is able to reduce the scale by doing the feature once only in the whole image and uses the crop on the lower parts of the sample [4, 5].

Faster R-CNN goes a step further using extruded materials used to create agnostic box suggestions, i.e., a single-release feature for the whole image, no external box suggestions. R-FCN is similar to Faster R-CNN, but feature harvesting is done on different types of enhancement efficiency. The advantage of the simplicity is that the YOLO model is faster compared to the Faster R-CNN and SSD which reads representation. This increases the error rate of the area and YOLO does not work well with images with a new feature rating but reduces the false rating. This will speed up real-time applications. This comes with the price of reduced accuracy [6–8].

SSD with MobileNet refers to the model in which the SSD and the type of SoftNet installer. Accurate tradeoff speed and many other modern systems for finding something that require real speed. Methods like the YOLO and SSD work faster, this often comes with a decrease in predictability, while Faster R-CNN models achieve higher accuracy but are more expensive to run. The cost per model speed depends on the application. SSD works compared to more expensive models like Faster R-CNN; in smaller images its performance is much lower [11].

## 3 Proposed Approach

The objective is to construct detection and counting model such that it will increase the efficiency of the loading/unloading process in trucks and also increment the efficiency of counting over human observation and lessen the chances of human forgery in the transportation industry. But before constructing the model, the need to study the present scenario and understand the factors or parameters that affect the statistics. The objective was pursued as given in the diagram.

To achieve the objective, the focus was put on data collection and analysis and model research. Also the detection model or implementation part is also a goal. The parameters and their prioritized values are decided only after proper analysis. For this, a dataset as large as possible is required. The discovery of the TensorFlow framework is used to build an in-depth learning network that helps solve object acquisition problems. This includes a collection of pre-defined retrained models trained in the COCO database, as well as the Open Pictures Dataset. These types can be used humbly if there is interest in categories on this data only. They are also useful for implementing models when trained in novel data.

### *3.1 Model Development*

Since the proposed model is based on custom object detection, there will be basic 6 steps to achieve the goal.

#### 3.1.1 Step 1: Problem Definition

This is the first step in finding an item. Defining a problem carefully requires an understanding of how things will be used, who needs the items, and how the adoption function fits into an organization that needs to predict. Job description, outsourcing, decision-making of a project, general discussion of trade-offs (accuracy vs speed), and setting a project codebase are the basic and fundamental requirements of a particular problem.

#### 3.1.2 Step 2: Gathering Information

There are at least two types of information required: (a) image data and (b) live camera feeds. Usually, it will be difficult to get enough live video data to be able to fit a good image capture model due to the large background noise. However, sometimes, older data will not be as useful due to changes in the upload method. For example, today, conveyor belts are more popular than handicrafts.

#### 3.1.3 Step 3: Training Data

Always start by labelling details. Training data are images collected as samples and interpreted to train deep neural networks. With the discovery of something, there are a lot of methods used to fix and quote data for training purposes. Many popular ways to distribute databases such as Pascal containing labelling tools.

#### 3.1.4 Step 4: Feature Extraction

Feature extraction is a basic step used by automated methods based on machine learning methods. Its main purpose is to extract useful features or important components from the data, which the computer can detect to calculate values in input images. An element is defined as the work of 1 or more dimensions that describe a particular asset (such as color, texture, or texture) of the whole image or layer or object.

#### 3.1.5 Step 5: Classification

The feature release feature provides a collection of records seen by a group of features x and a class label y. Aimed at describing a class model that includes a recording class label. Image classification follows steps such as pre-processing, component fragmentation, feature extraction, and segmentation.

#### 3.1.6 Step 6: Testing Data

There is test data within the file which holds to 30% of the full dataset. This test data isn't being employed for training the model. Even after training the model, it is required to check the model on some dataset. It's very significant to understand the accuracy of the full model and confidence score of a selected test run on a trained model.

## 4 Proposed Methodology

The goal of the sack counting system is to process the image frames passed to it via the image processing algorithm mentioned in the above diagram. The engine is trained as such, that, it is processes the image frames found in the frame folder and predicts the sack using a predictive algorithm and returns the live count of the sacks loaded on to the truck. The engine used the OpenCV and the image frames are fetched from the live recording from the CCTV camera recording the video. The external interface of this engine begins with the collection of frames saved in a .csv folder (Fig. 1).

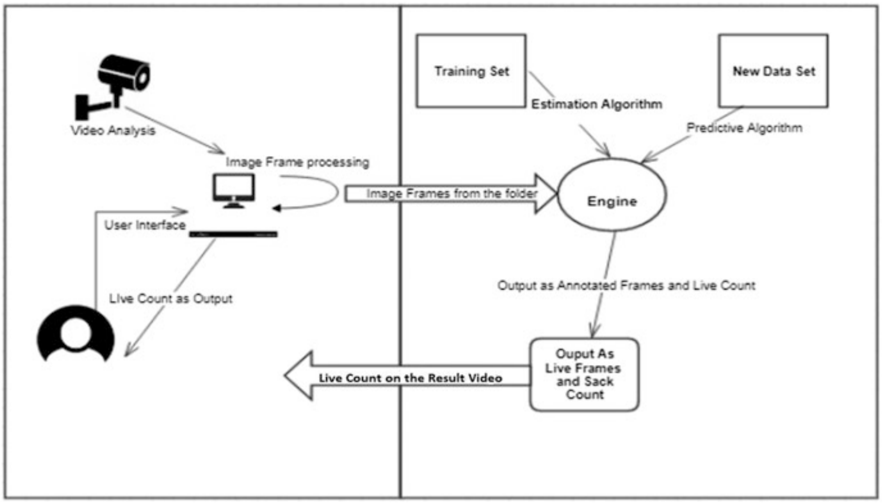


**Fig. 1** Architecture diagram of sack count system

The frames are extracted through a live video stream of Camera installed at the work area. Then the OpenCV engine takes the frames one by one from the folder and starts analyzing through a command-line interface. The engine then identifies the correct frames and increments the counter, now the frames become annotated. Finally, the counter and annotated frames, the desired result will be the output video. The annotated result video will become the output. The internal interface will be decomposed into two parts: Train and Test.

The train part will be used to train the model for detecting the frame with a sack, initially making of the dataset is done by own and divided randomly into training and validation sets. It contains images frames along with the label of sack marked on that frame using labelling annotation. Here, the engine will call the main function which in turn calls the train function which first initializes the parameters randomly and updates weights at each iteration up to some minimum cost. The model hyperparameters according to the variance and bias provided by the model are tuned. It is a continuous process to make a model robust to new images and provide more accuracy. This process will train the model, and final parameters can be used to predict the new frame.

In the test part or operational phase, the main function will be called, which refers to predict function. The predict function will check the image frame according to the trained model parameters. If the sack is present in an image frame, then the result would be there, in the form of an output result video which will show the number of sacks loaded/unloaded in truck and time and date of the loading/unloading in truck.

## *4.1 System Design*

1. Data Flow Design

The data flow diagram (DFD) shows the flow of information for any process or system. Data flow diagrams represent systems and processes that will be difficult to define in textual format (Fig. 2).

## *4.2 Sequence Diagram*

A sequence diagram shows the object interactions that are arranged in time sequence. It would describe the classes and objects which are involved in the sequence of messages and exchange between the objects needed to carry out the functionality of the structure (Fig. 3).

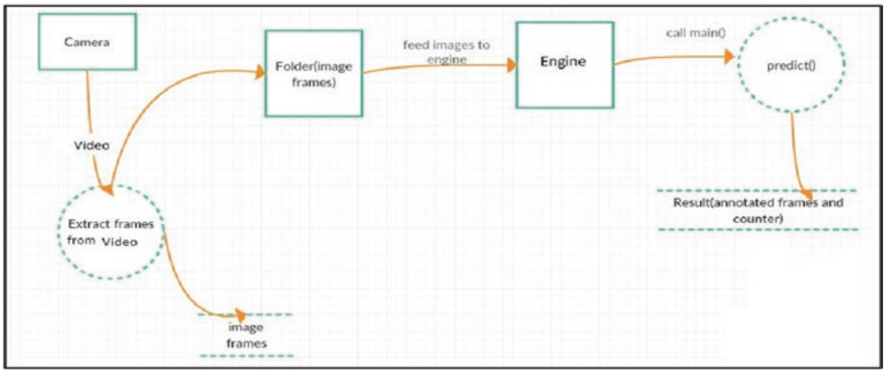


**Fig. 2** Data flow diagram of the proposed method

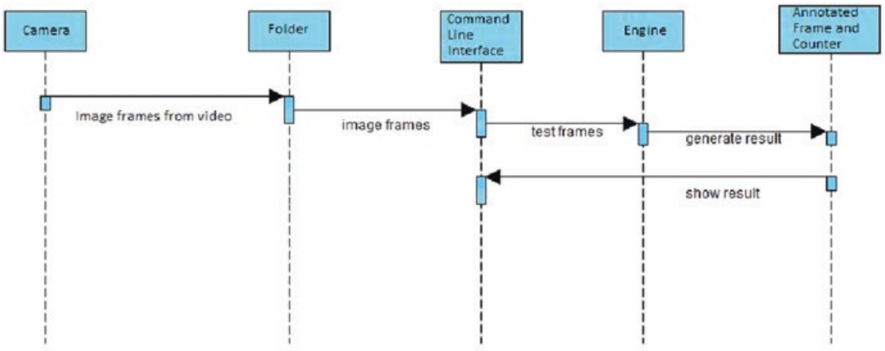


**Fig. 3** Sequence diagram of the proposed approach

## 5 Implementation Details

### *5.1 TensorFlow framework*

TensorFlow is an open-source framework used to integrate and build great machine learning programs. It is created by the Google Brain team. It incorporates mechanical and in-depth learning methods as well as neural network models and algorithms and makes them useful in the form of a standard platform. It uses Python as a programming language that provides a simpler and better complete API for building applications using a framework and also aids in the development of those applications in greater C ++ functionality. TensorFlow helps with prediction of production at a certain level, with the same models used to train a deep neural network. It can train and operate deep neural networks. For example, neural networks used for image recognition, handwriting digitization, duplicate neural networks, embedding words, sequential machine translation models, etc.

Python language is supported by TensorFlow to provide all framework functions. Python is a simple structured language. It is easy to read and work on and offers useful options for explaining how high-end manufacturers can be integrated. Nodes and tensors are Python objects, and the provided Python applications are TensorFlow applications themselves.

In Python, real mathematical functions are not performed. C ++ phones that are known for high-performance work with conversion libraries available for language processing, as well as PDE (which is part of the separate classification) used.

TensorFlow allows developers to create dataflow graphs. Dataflow graphs are defined as structures that show how data moves across a graph, or processing nodes in a series. Mathematical performance is represented by nodes in the graph. Each edge between the areas is a multi-distance data system called tensor.le in the TensorFlow framework. Advanced scams are provided by Python which simply directs traffic between pieces and connects loopholes at intersections.

TensorFlow applications can be distributed to any target and user-friendly, be it machine, cloud collection, iOS and Android devices, or CPUs or GPUs. If you use Google's own cloud, you'll be able to use TensorFlow in the Google Silicon TensorFlow Processing Unit (TPU) to keep up the speed.

Output models of the TensorFlow framework, however, will be used on any device that can be used to provide prediction. TensorFlow 2.0 analyzes the framework in some ways that support user feedback, to make it easier to find that (e.g., by using the simple Keras API for training) and to do more. Distributed training is very complex to run due to the replacement API, and support for TensorFlow Lite enables us to deploy models in a large platform style. However, the code written for previous versions of TensorFlow must be rewritten only occasionally, and sometimes significantly so that the maximum benefit of the latest TensorFlow 2.0 features can be established.

One of the great advantages of TensorFlow is that it offers the advantages of machine learning development. Instead of managing the nitty-gritty details of using algorithms, or determining the appropriate ways to capture the first result of a task in another installation, the developer can work specifically on the general concept of the application. TensorFlow takes care of the small print behind the scenes.

Also, TensorFlow offers one more benefit for developers who need to fix a problem and take TensorFlow details. In the framework, there is a mode of eager use that allows one to test and modify the performance of each graph especially and without stitching, instead of naming the graph as one opaque object and testing it all at once. And the TensorBoard visualization Suite allows one to view and select graph-driven interactive dashboard.

TensorFlow also received a chance for Google support. Google has not only allowed for faster growth after the project but also has created many important TensorFlow methods that make it much easier to use and operate. For example, the abovementioned TPU silicon for faster performance in the Google cloud, your browser and the desired size for the frame, the hub for online frame sharing, and much more.

## 5.2 Faster R-CNN Algorithm

R-CNN was launched in 2014 and has attracted a lot of interest in the computer vision community. The whole idea behind R-CNN is to implement a search engine selected to propose 2000 Region-and-Interest (ROI), which is then fed into the Convolutional Neural Network (CNN) to collect features. These features have been used to classify photographs and their object boundaries using SVM (support vector machine) and regression methods. For a more detailed explanation, see this section. And this approach was quickly followed by R-CNN, which became a faster and better way to identify an object. Fast R-CNN uses a ROI pooling system that shares features throughout the image and uses a modified type spatial pyramid pooling method to efficiently capture features over time. In the case of fast R-CNN, it is still slow because it has SS to handle, which is computationally very slow. The test time to test Fast R-CNN takes from 47 s to 0.32 s and 2 s to generate 2000 ROIs. It adds up to 2.3 s per image (Fig. 4).

The shortcomings of the above two algorithms paved the way for researchers to quickly come up with R-CNN, where the test time for each image with field resolutions was only 0.2 s. This is due to the latest approach given the completely different model used for end-to-end training.

There are two proposals for fast R-CNN, such as field proposal networks (RPNs) used to generate field proposals and networks that use these found resolutions to identify objects. The big difference here with Fast R-CNN is that the latter fields use selective search to obtain proposals. The time and cost of building field resolutions is much lower in RPNs than in selected searches, as RPNs share very important calculations with the detection network. However, RPNs call field boxes anchors and propose them as objects.

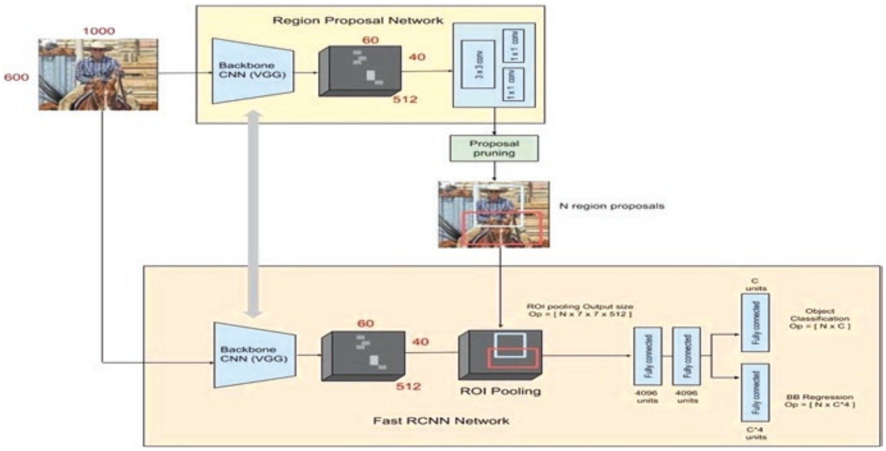


**Fig. 4** Faster R-CNN algorithm

The Faster R-CNN Object Detection Network feature is designed with an extraction network that can often use a blocked traditional neural network. After, this two subnetworks can be trained. The primary may be the Region Proposal Network (RPN), which is accustomed to generating object conclusions. The latter is therefore used to estimate the specific class of a given article. The primary variant for fast R-CNN is RPN, which is added after the final composite layer. It is often trained to provide field conclusions without the need for external mechanisms such as selective search. ROI pooling can be used as an upstream classification and bounding box resistor similar to the one used in Fast R-CNN.

## *5.3 Inception Model*

The beginning may be the deeply controversial neural specification introduced in 2014. It won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC14). It was mostly developed by Google researchers.

In convolutional neural networks (CNNs), the outer part of the work is performed at the appropriate layer for use between the most common options (1 × 1 filter, 3 × 3 filter, 5 × 5 filter, or maximum-pooling). We all want to find the right local structure and replicate it spatially (Fig. 5).

As each of these inception modules is stacked on top, their output correlation statistics vary: as the properties of the upper extraction are captured by the higher layers, their spatial concentration of 3 × 3 and k reduction is estimated, indicating the ratio. 5 × 5 resolutions increase as we move to higher layers.

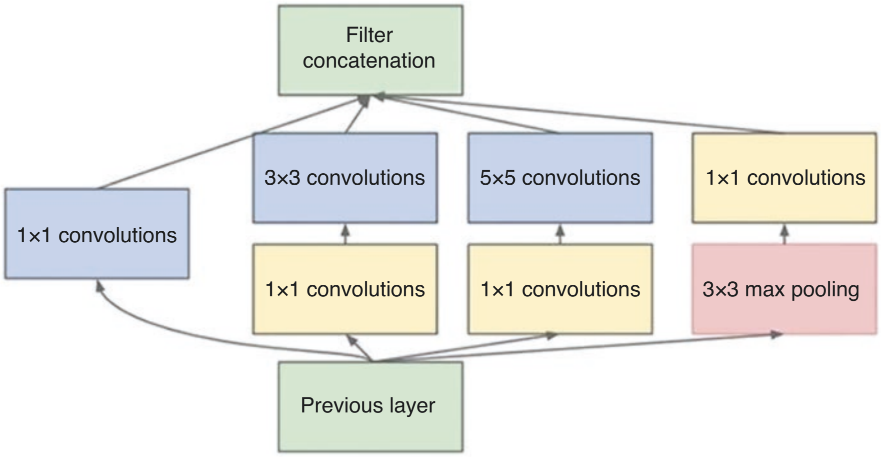


**Fig. 5** Inception model

However, the computational cost of such an answer would increase enormously. For this reason, within the figure, diffusion reduction is used as diffusion reduction methods by 1 × 1 convolution.

The installation module aims to act as a “multi-level feature extractor” by calculating the 1 × 1, 3 × 3 and 5 × 5 additives in the homogeneous modules of the network – the output of those filters being then stacked together. Channel size and before feeding in the bottom layer of the network.

The original version of this architecture was called GoogleNet, but later versions were called Inception VN, where n refers to the version number entered by Google. The Inception V3 architecture included in the Kears Core comes from a later publication by Szegedi. Inception Architecture for Computer Vision (2015) proposing an update of the Inception module to further increase image net classification accuracy. Initially the V3 weighed 93 MB, which is smaller than both VGG and ResNet.

### 5.3.1 Data Collection

The classifier is built in taxonomy to identify sacks. Dataset is a very important thing in creating taxonomy. This may be the basis for our classification, on which object detection ceases. Various and varied images containing objects are collected. Then the directory name image inside the research directory is created. Now, 80% of the photos are stored in the train directory and 20% of the images are stored in the test directory inside the pictures directory. We have collected 174 images in the train directory and 35 images in the test directory (Figs. 6 and 7).

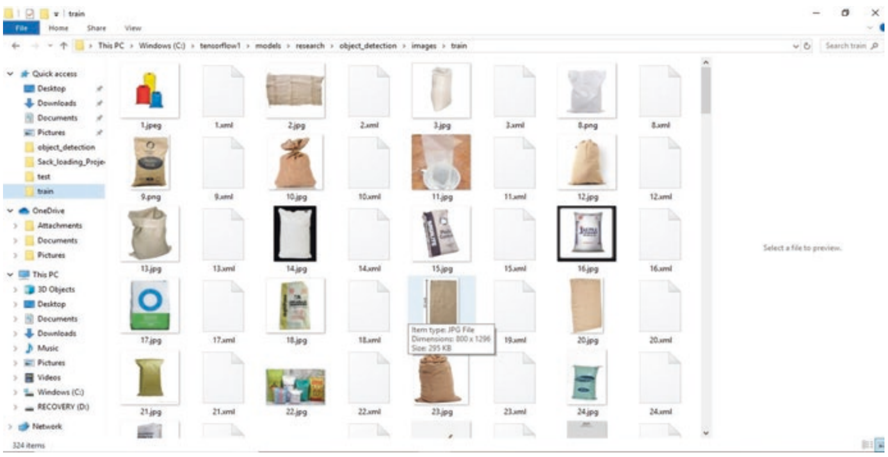

This PC › Windows (C:) › tensorflow1 › models › research › object_detection › images › train
324 items
Select a file to preview.

Fig. 6 Train directory

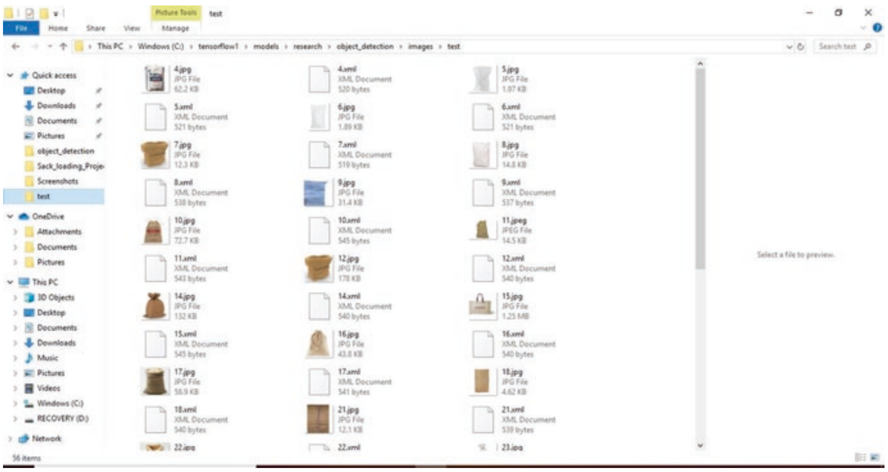

This PC › Windows (C:) › tensorflow1 › models › research › object_detection › images › test
56 items
Select a file to preview.

Fig. 7 Test directory

### 5.3.2 Labelling the Dataset

Open the Labelling tool and start drawing rectangular boxes on the image where the subject is. Label them with the appropriate name as shown in Fig. 8. Save each image created by labelling an xml file with the corresponding image name as shown in Fig. 9.

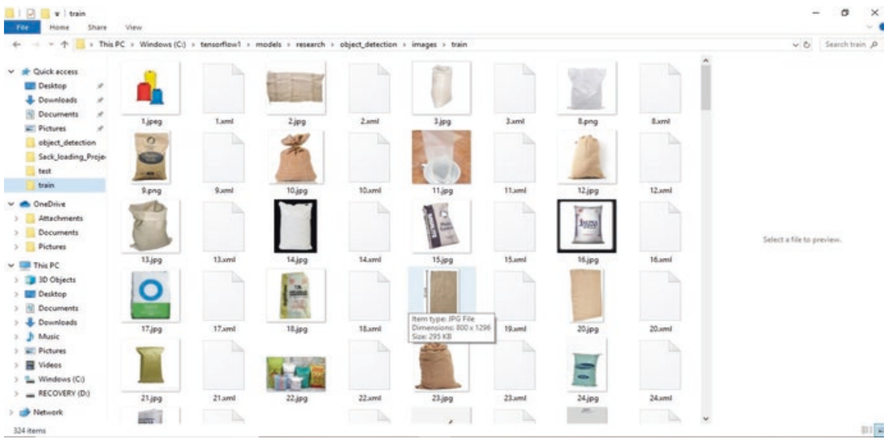
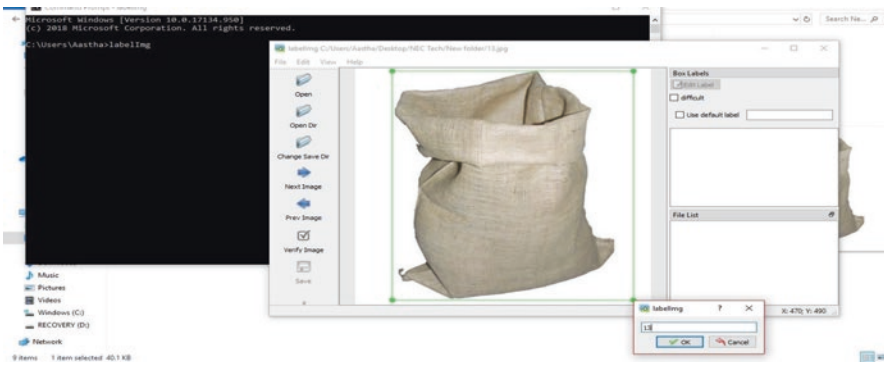


**Fig. 8** Train directory with xml file



```
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Aastha>labelImg
```

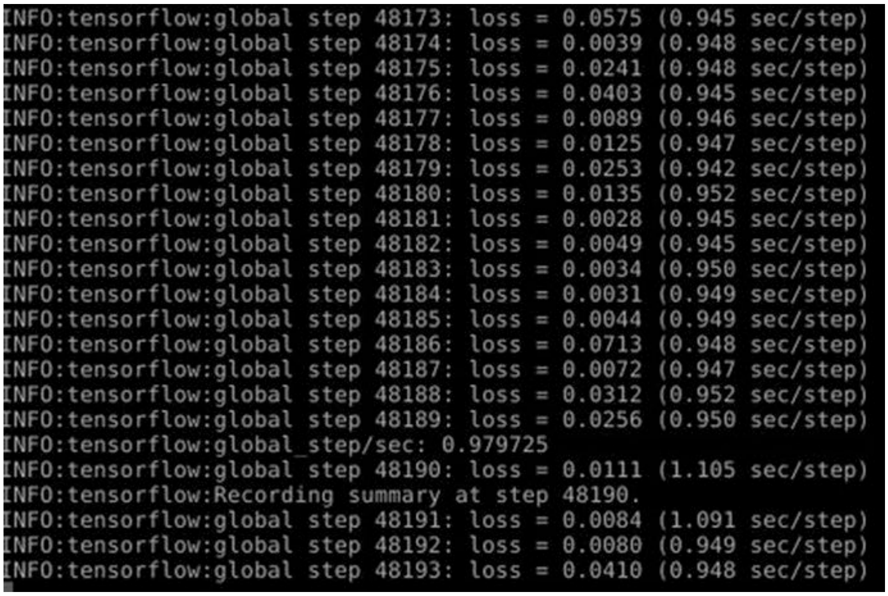
**Fig. 9** Labelling tool

### 5.3.3 Generating TensorFlow Records for Training

For this step, we want to create TFRecords that can be displayed as an input file to train the subject detector. Xml_to_csv.py and Gener_tfrecord.py codes are given. Create test.csv and train.csv files in the xml_cs_csv.py images folder. Then, create files for recording in TensorFlow by executing the appropriate commands from the Object_Detection folder.

### 5.3.4 Configuring Training

Create a replacement directory called Training in the Object_Detection directory. Use the text editor to create the restore file and place it as a label map in the training directory. The label tells the map instructor what each object is by defining the



```
INFO:tensorflow:global step 48173: loss = 0.0575 (0.945 sec/step)
INFO:tensorflow:global step 48174: loss = 0.0039 (0.948 sec/step)
INFO:tensorflow:global step 48175: loss = 0.0241 (0.948 sec/step)
INFO:tensorflow:global step 48176: loss = 0.0403 (0.945 sec/step)
INFO:tensorflow:global step 48177: loss = 0.0089 (0.946 sec/step)
INFO:tensorflow:global step 48178: loss = 0.0125 (0.947 sec/step)
INFO:tensorflow:global step 48179: loss = 0.0253 (0.942 sec/step)
INFO:tensorflow:global step 48180: loss = 0.0135 (0.952 sec/step)
INFO:tensorflow:global step 48181: loss = 0.0028 (0.945 sec/step)
INFO:tensorflow:global step 48182: loss = 0.0049 (0.945 sec/step)
INFO:tensorflow:global step 48183: loss = 0.0034 (0.950 sec/step)
INFO:tensorflow:global step 48184: loss = 0.0031 (0.949 sec/step)
INFO:tensorflow:global step 48185: loss = 0.0044 (0.949 sec/step)
INFO:tensorflow:global step 48186: loss = 0.0713 (0.948 sec/step)
INFO:tensorflow:global step 48187: loss = 0.0072 (0.947 sec/step)
INFO:tensorflow:global step 48188: loss = 0.0312 (0.952 sec/step)
INFO:tensorflow:global step 48189: loss = 0.0256 (0.950 sec/step)
INFO:tensorflow:global_step/sec: 0.979725
INFO:tensorflow:global step 48190: loss = 0.0111 (1.105 sec/step)
INFO:tensorflow:Recording summary at step 48190.
INFO:tensorflow:global step 48191: loss = 0.0084 (1.091 sec/step)
INFO:tensorflow:global step 48192: loss = 0.0080 (0.949 sec/step)
INFO:tensorflow:global step 48193: loss = 0.0410 (0.948 sec/step)
```

**Fig. 10** Training model

mapping of the advanced name to the class ID number. Now, add the content to class-map.pbtxt, then switch to the format to create a label map for your taxonomy. The label map ID number should be as defined in the Generate_tfrecord.py file. We need a model algorithm to train our classification. During this project, we are visiting to use the faster_rcn_inception model. TensorFlow's Object Detection API includes a large number of models. Now, open the file using the text editor and make the necessary changes to the fast_rcnn_inception_v2_pets.config file saved in the directory.

### 5.3.5 Training Model

At the situation object_detection/legacy/find the file train.py. Open the object detection directory and copy the train.py file and paste it within the same directory. Run the subsequent command to begin training the model in the object detection folder itself. It takes around 1–2 min to start the setup before the training begins. The training starts, and it looks like given in Fig. 10.

## 6 Conclusion

The work done so far is concentrated on the collection image data and labelling from and training the model for object detection in an image. Also the focus was on different algorithms that are there and their comparative study for analysis of video data. The detection model will take these resulting parameters as input and produce a prediction. The goal to convert this analysis into an accurate detection and counting using these object detection and tracking algorithms has been achieved.

The framework used, that is the TensorFlow framework, in this project for the application of object count can be further extended and optimized through various means. It can make simple but significant modifications in the model while training and during data pre-processing like increasing the training size, more epochs in training, and bigger size images for improving the accuracy. Also we can implement this whole idea on mobile devices using MobileNet (included in TensorFlow framework) which provides better accuracy.

## References

1. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, Speed/accuracy trade-offs for modern convolutional object detectors. arXiv preprint (2016) arXiv:1611.10012
2. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), pp. 779–788
3. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A.C. Berg, Ssd: Single shot multibox detector, in *European Conference on Computer Vision*, (Springer, 2016), pp. 21–37
4. B. Galitsky, A content management system for Chatbots, in *Developing Enterprise Chatbots*, vol. 1, (Springer, 2019), pp. 253–326
5. N. Ali, M. Hindi, R.V. Yampolskiy, Evaluation of authorship attribution software on a chatbot corpus, in *International Symposium on Information, Communication and Automation Technologies*, (2011)
6. E. Go, S.S. Sundar, Humanizing chatbots: The effects of visual, identity and conversational cues on humanness perceptions. Comput. Hum. Behav. **97**, 304–316 (2019)
7. P. Singh, H. Khatter, S. Kumar, Evolution of software-defined networking foundations, in *Evolution of Software-Defined Networking Foundations for IoT and 5G Mobile Networks*, vol. 1, (2021), pp. 98–112
8. H. Khatter, B.M. Kalra, A new approach to Blog information searching and curating, in *Proceedings to Sixth International Conference on Software Engineering CONSEG 2012, Indore, India*, (2012), pp. 1–6
9. H. Khatter, A.K. Ahlawat, Analysis of content curation algorithms on personalized web searching, in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*, (New Delhi, 2020), pp. 1–4. https://doi.org/10.2139/ssrn.3563374
10. H. Khatter, M.C. Trivedi, B.M. Kalra, An implementation of intelligent searching and curating technique on Blog Web 2.0 tool. Int. J. U E-Serv. Sci. Technol. **8**(6), 45–54 (2015)
11. H. Khatter, A.K. Ahlawat, An intelligent personalized web blog searching technique using fuzzy-based feedback recurrent neural network. Soft Comput. **24**(12), 9321–9333 (2020). https://doi.org/10.1007/s00500-020-04891-y