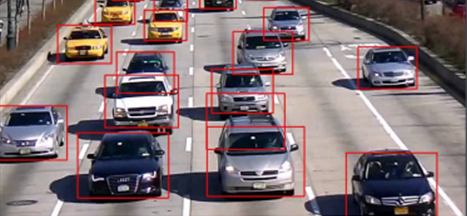
|  |
| --- |
| **Deep Learning Based Car Identification**  ***Automotive Surveillance, Object Detection & Localisation*** |
| July 2022Team: July 21B- G1-CV2(Mentor: Shyam Muralidharan)Team Members: [**Premjeet Kumar**](https://www.linkedin.com/in/premjeet-kumar/) **,** [**Hari Samynaath S**](https://www.linkedin.com/in/harinaathan/)**,** [**Veena Raju**](https://www.linkedin.com/in/veena-raju-1b16b513)**,** [**Javed Bhai,**](https://www.linkedin.com/in/javedbhai/)[**Surabhi Joshi**](https://www.linkedin.com/in/surabhi-joshi-4452788/) |



**Table of Contents**

[**ABSTRACT**  **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033820)

[**INTRODUCTION** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033821)

[**Problem Statement:** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033822)

[**Objective** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033823)

[**Data sources** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033824)

[**EXPLORATORY DATA ANALYSIS** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033828)

[**Training Data Frame Visualization:** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033829)

[**Visualization of different classes]** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033830)

[**MODEL BUILDING** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033831)

[**Deciding Models and Model Building** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033832)

[**Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033833)

[**Training the model** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033834)

[**Predicted Bounding Boxes Versus Ground Truth Boxes Using The Validation Dataset:** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033835)

[**Improve model performance** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033836)

[**Next steps** **Error! Bookmark not defined.**](file:///F:\Capstone_group_cv3_inter_report_W2024.7.docx#_Toc43033837)

CoMPUTER VISION

### **Abstract**

TBD

## Introduction

Object detection is a [computer vision](https://www.mygreatlearning.com/academy/learn-for-free/courses/computer-vision-essentials?gl_blog_id=18067) technique in which a software system can detect, locate, and trace the object from a given image or video. The special attribute about object detection is that it identifies the class of object (person, table, chair, etc.) and their location-specific coordinates in the given image. The location is pointed out by drawing a bounding box around the object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the performance of the algorithm used for detection.

### **Problem Statement:**

Computer vision-based models can be effectively used for object detection in real time. Ability to easily identify a moving vehicle on road through camera can go a long way in automating road supervision and surveillance for various business and law enforcement purposes. Further, these models can be integrated with other network systems for generation of appropriate actions triggers. Designing and building a computer vision model which can be used as vehicle recognition predictive models or car classification models can provide an effective solution for various vehicle detection application. V

### **Objective**

To design a deep learning-based car identification model that can be deployed to automate detection, identification and surveillance of cars on road for various business and law enforcement purposes. The model will enable identification of car moving on the road by a camera as make, type, model and OEM.

o build n algorithm to detect a visual signal for pneumonia in medical images. Further algorithm needs to automatically locate lung opacities on chest radiographs

### **Data sources**

The car detection model will be prepared using The Stanford Cars dataset, which is developed by Stanford University AI Lab specifically to create models for differentiating car types from each other.

The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

Data description:

Train Images: Consists of real images of cars as per the make and year of the car.

Test Images: Consists of real images of cars as per the make and year of the car.

Train Annotation: Consists of bounding box region for training images.

Test Annotation: Consists of bounding box region for testing images.

The useful data to create the model is available in three files including two zipped folder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S No. | File name | Format | Size | Details |
| 1 | Car Images | . zip |  | The .zip folder includes two sub folders Train and Test. Each of these subfolders have 196 class folder with .jepg images of folders |
| 2 | Annotations | . zip |  | CSV files in two folders train and test with bounding box coordinates |
| 3 | Car names & make | .CSV |  | Single CSV file with car name and makes indeed with 196 car classes |

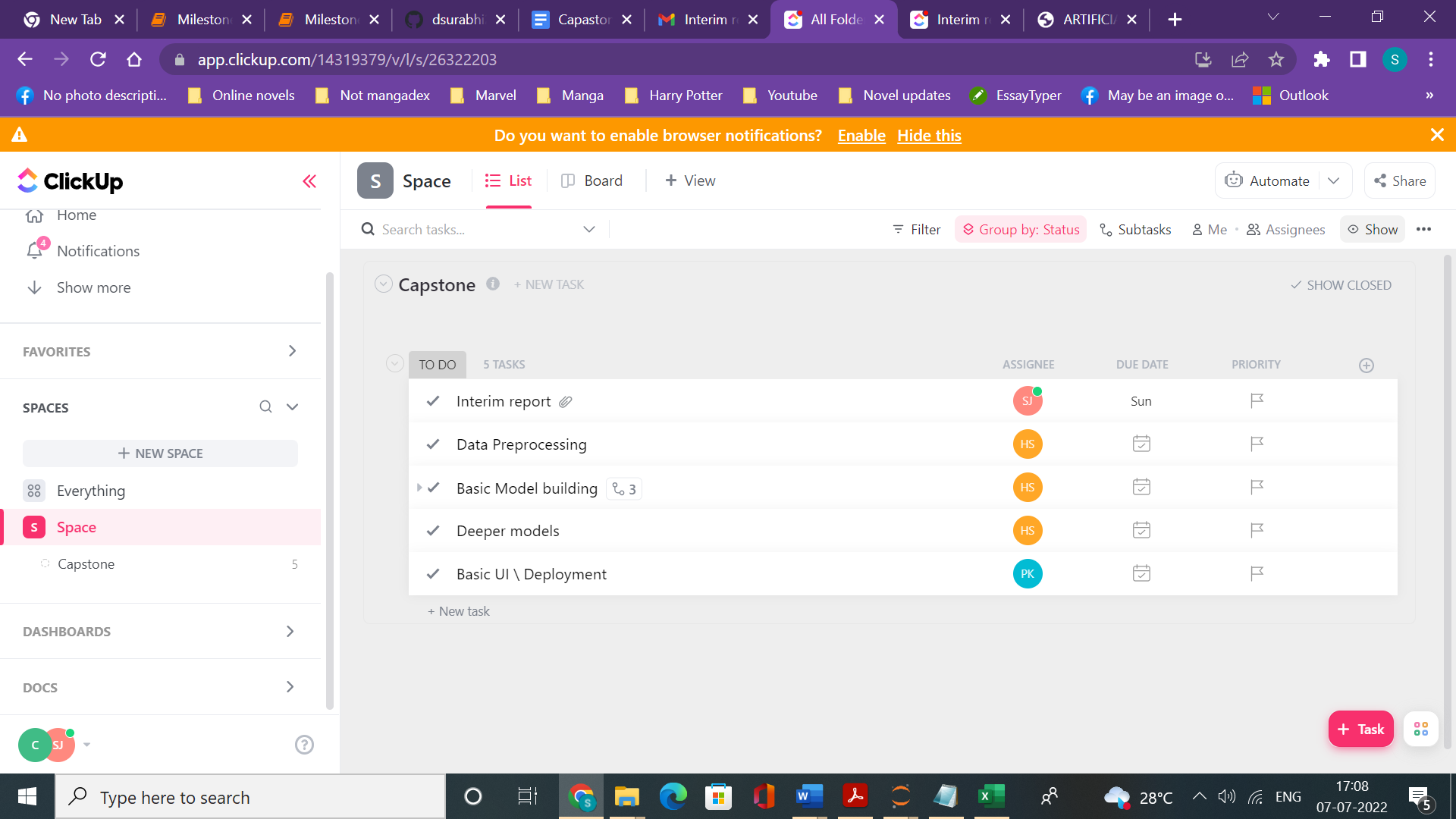
### **Problem Approach**

### This project requires a constant collaboration and sharing along with task differentiation and allocation amongst the team members for timely execution and delivery of interim and final targets . The team under the able guidance of mentor Mr. Shyam Muralidharan , has created a GitHub repository (<https://github.com/pkrsingh2/Detect-car-model-using-computer-vision>) .

### **Github repository for collaborative sharing and managing the work flow**

### 

In Click Up main activity dashboard created and activities and task progress monitored.



### Further the team communicates and coordinates on daily basis for effective sharing and learning over the entire project life cycle and managing the work flow.

### The work flow includes

### The key steps for project implementation includes setting up a pipeline for data extraction , exploratory data analysis to understand the available data, preprocessing , model building , model assessments , model improvement for best model selection followed by deployment

### 

**CAR  
CLASSIFIER**

### **In the ne**x**t section we detail the steps and insights for data extraction and exploration**

### Google Cloud Platform created

### **Data Extraction and Exploration**

### Necessary libraries were imported for analysis and the data files were imported using python note book. The data extraction included three steps

### 1.Reading the provided .CSV files and image files

### 2.Reviewing the name lengths and format in the CSV files for identifying different features i.e. OEM, Car type, year of make

### 3.Checking for any inconsistencies and missing data

### The car name & make CSV file provides names of 196 classes of cars for which images are available for model training inde**x**ed with class number. The name provides a combined information in four categories car manufacturer, model name , car type and Car make year. The names and name length are initially studied

### 

### 

### We find that full name length range between 4 to 7 words in the data base with information provided for the four features OEM, Type, Model and make year in the respective order . Through multiple steps of data sorting we could segregate the features of Car entries in four different columns

### As the key feature categories were extracted the data became comprehensible to carry a detailed exploratory data analysis

#### 

#### Exploratory DATA ANALYSIS

### We do an overall review of heterogeneity in the data by understanding number of unique values for each variable

### 

### We find that data includes 49 distinct OEM’s with 173 different car models of 23 different type of cars. The data spans for 16 unique time periods

### We now explore the frequency distribution and data distribution of various categories of cars WRT to OEM’s , Type of cars , and year of manufacturing and plot the frequencies

### **Number of Models for each Car Manufacturer ( OEM)**

### 

### The data consists of 22 models for Chevrolet followed by Dodge (15), Audi(14), BMW(12) , Ford(12) and Hundai (11) which have more then 10 models in the data thus higher representation that significantly higher RAM , MADA etc which have only one model in the data

### **Number of Models for each Car Type**

### 

### Most represented car type in data are Sedans which have 46 models in the current dataset

### **Number of Models for each make year**

### 

### From the perspective of data distribution highest data frequency for make year in 117 , which emerges as a predominant bias in the existing dataset . From the above three parameters we get an insight that there is a high probability of instrinsic biases in terms of predominant OEM, type and year of manufacturing for the data .

### We further explore this findings by going deeper into understanding the data distribution interms of available images for each category 196 classes of cars in the data set . For doing we do analysis in two steps .

### **Data Distribution in terms of individual image counts**

### Step one we try to evaluate an median counts for images for training for each class as that will be crucial in determining the training biases for the model

### We determine the distribution support of each class and find that for most classes the total images available lie between 40-50 instances for training

### 

### Number of images in each OEM

### 

### Number of images in each OEM

### 

### Number of images in each make year

### 

### In the above analysis the frequency distribution of images closely follows the class distribution with Chevrolet, sedan and 2012 make year emerging as the predominant data biases

### We further evaluate combined frequencies for various models as the inputs would be going in terms of 196 training classes (multiclass training)

### Combined distribution of images wrt car type , OEM and car make year

### 

### We find that although Chervelotee has maimum images highest number of images for training are available for other OEMs like GMC, Hyundai , Jeep etc . Further highest number of training images are available SUVs with in a class thus Sedan alone do not form predominant image class at class level . Further the 2012 emerges as the predominant make year even in this combination analysis .

### Further we try to understand distribution for lesser represented classes in terms of images

### 

### We find that lesser instances are found for cars like FIAT , May Batch , Rolls Royce but least image number in 28 which is for unknown type cars .

### The above combination analysis highlight that

### There are inherent biases in terms of car make year so the model that would design would be appropriate for lower range in terms of make year in terms of efficiency

### **Data Preprocessing**

The acquired data from the real world is usually messy and come from different sources. To feed them to the machine learning or neural network-based model, they need to be standardized and cleaned up.

Preprocessing is more often used to conduct steps that reduce the complexity and increase the accuracy of the applied algorithm. As writing a unique algorithm for each of the condition

in which an image is taken would be very cumbersome and resource constraining thus, when an image is acquired, its first converted into a form that allows a general algorithm to solve it.

### **A manual survey of training and test images reveal that images have different backgrounds and resolution. Therefore, use of bounding boxes and image size normalization becomes essential**

### **Snap shot Training Images**

### 

### **Snap shot Training Images**

### **Snap shot Training Images**

### 

### **The data preprocessing included**

### **Image** **size Normalization**

### **Augmentation of the Bounding boxes**

### **Image size Normalization**

### **We initially compute image size and print image dimension in a separate column**

### 

#### 

**The size of the images vary from as large as 210 Mega pixels to 4.5 kilo pixels**

**Resizing Images :** Resizing images is a critical preprocessing step in computer vision. Machine Learning models train faster on smaller images and they need images of same size as input.

**Some of the Best Practices**

1. To decide on what should be the size of the images, a good strategy is to employ progressive resizing. eg; we can start with all images resized to the smallest one.

2. Progressive resizing will train an initial model with very small input images and gauge performance. We can use those weights as the starting point for the next model with larger input images.

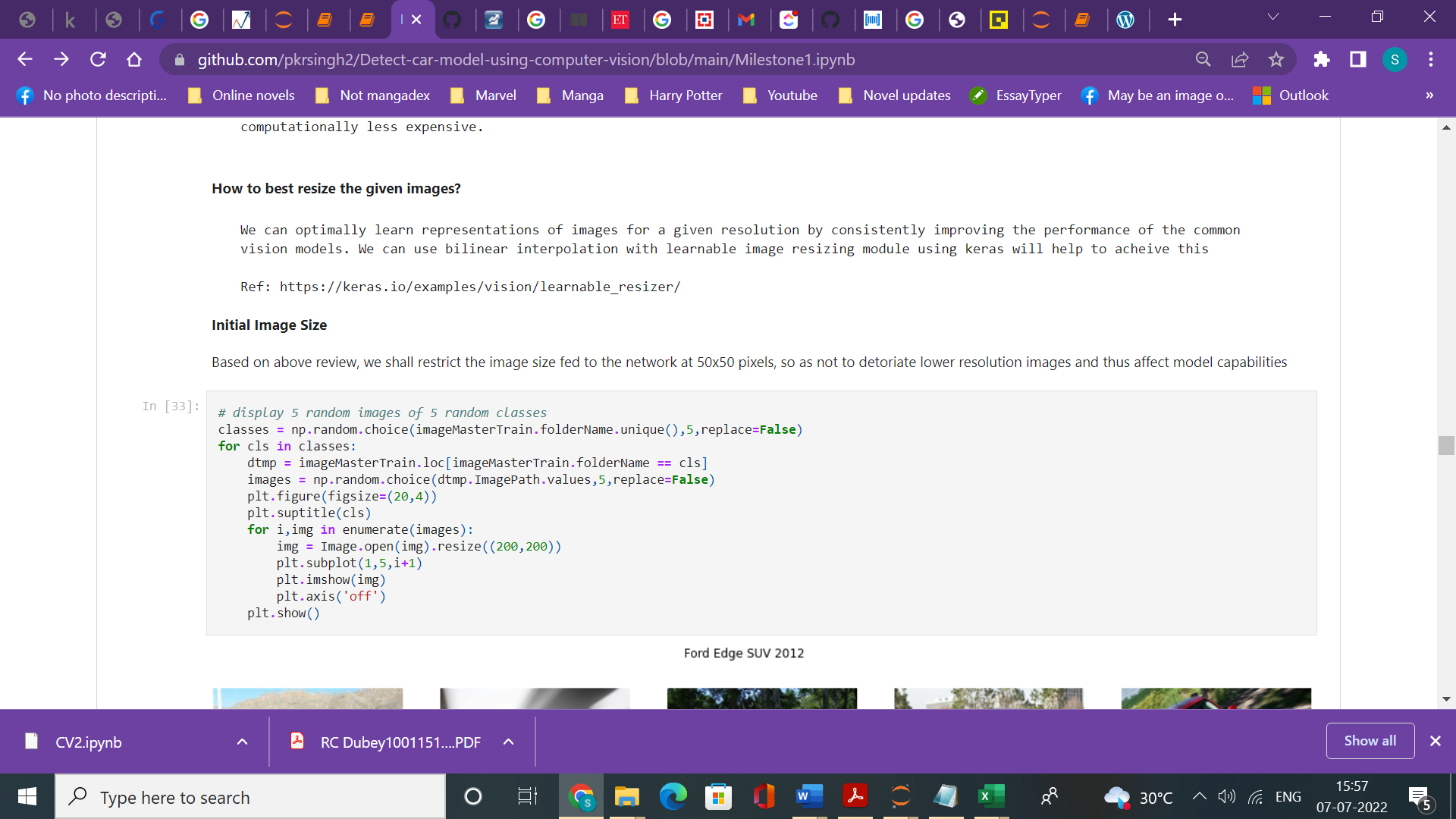
3. Downsizing larger images to match the size of smaller images is often a better bet than increasing the size of small images to be larger.

4. In general, it is safer to maintain the raw image aspect ratio and resize proportionally.

5. Make use of image resizing methods like interpolation so that the resized images do not lose much of their perceptual character.

**Initial Image Size**

Based on above review, we shall restrict the image size fed to the network at 50x50 pixels, so as not to deteriorate lower resolution images and thus affect model capabilities

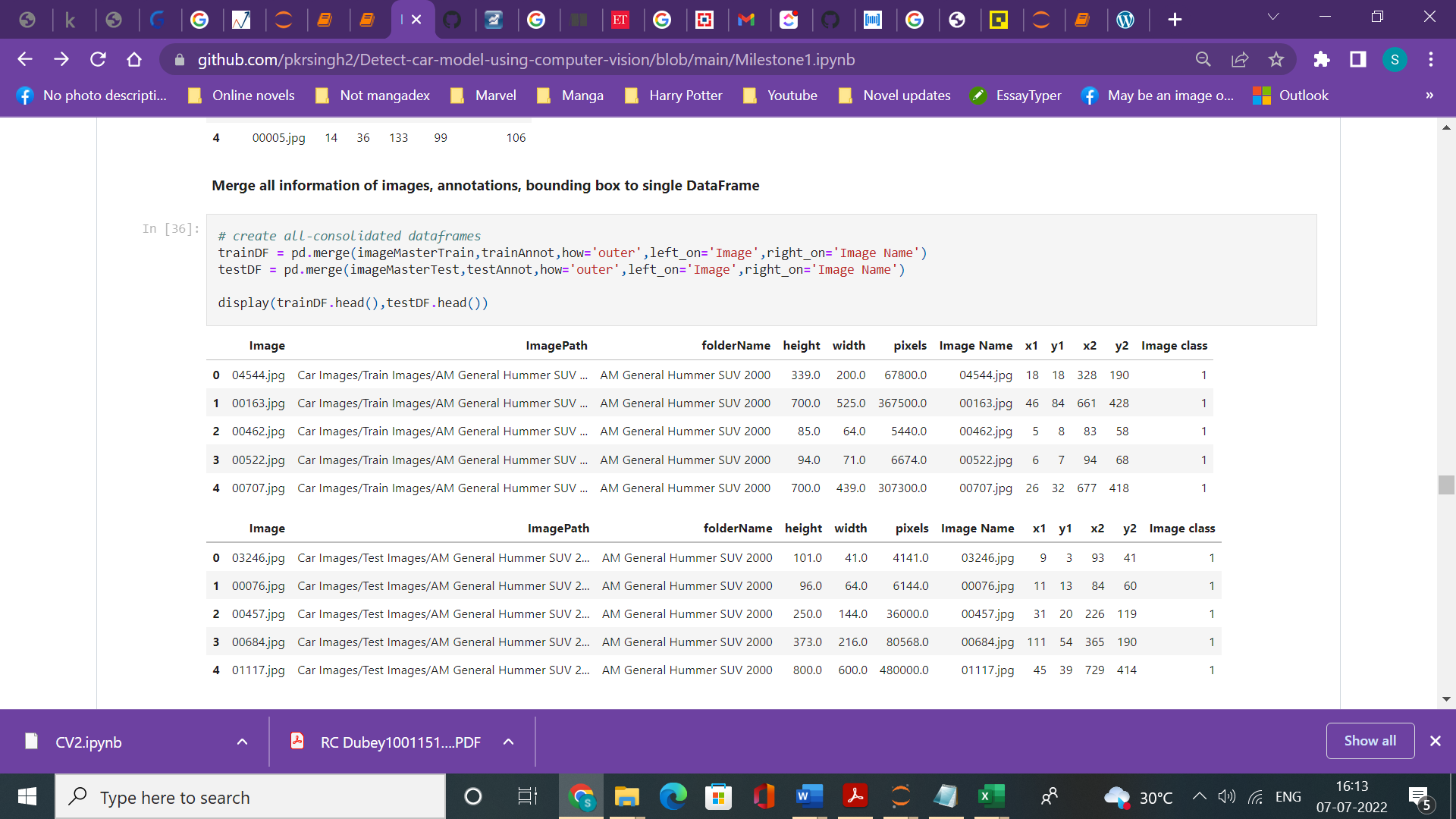


#### 

After converting all images to same sie we still have the images with different backgrounds and resolutions . We now use the already available annotations to create bounding boxes.

**Adding Bounding Boxes**

1. **Step 1 : Merge all the information of images, annotations, bounding box to single Data Frame**



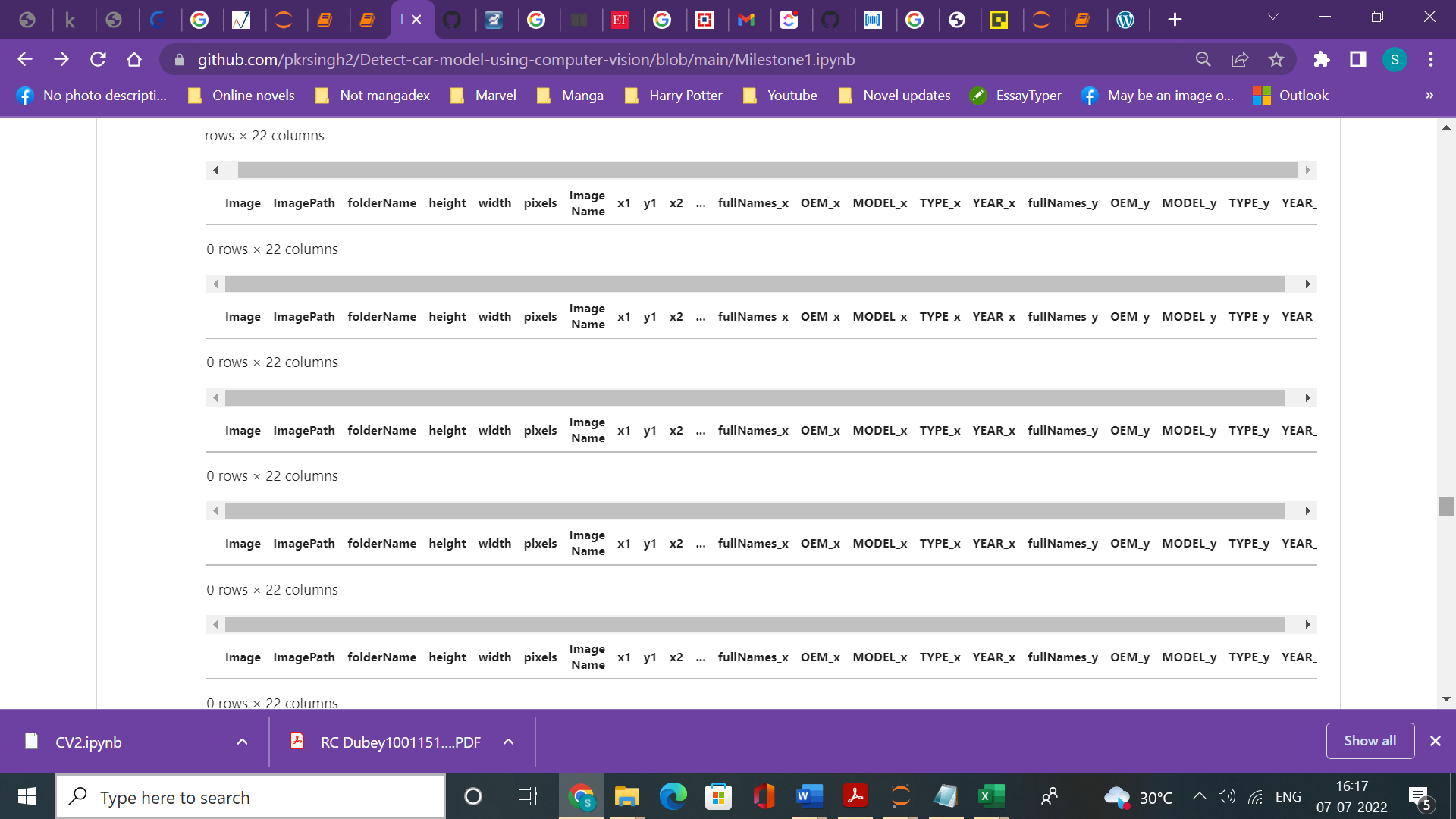
1. **Step 2: Merge OEM, MODEL, Type, Year with the above data frame**



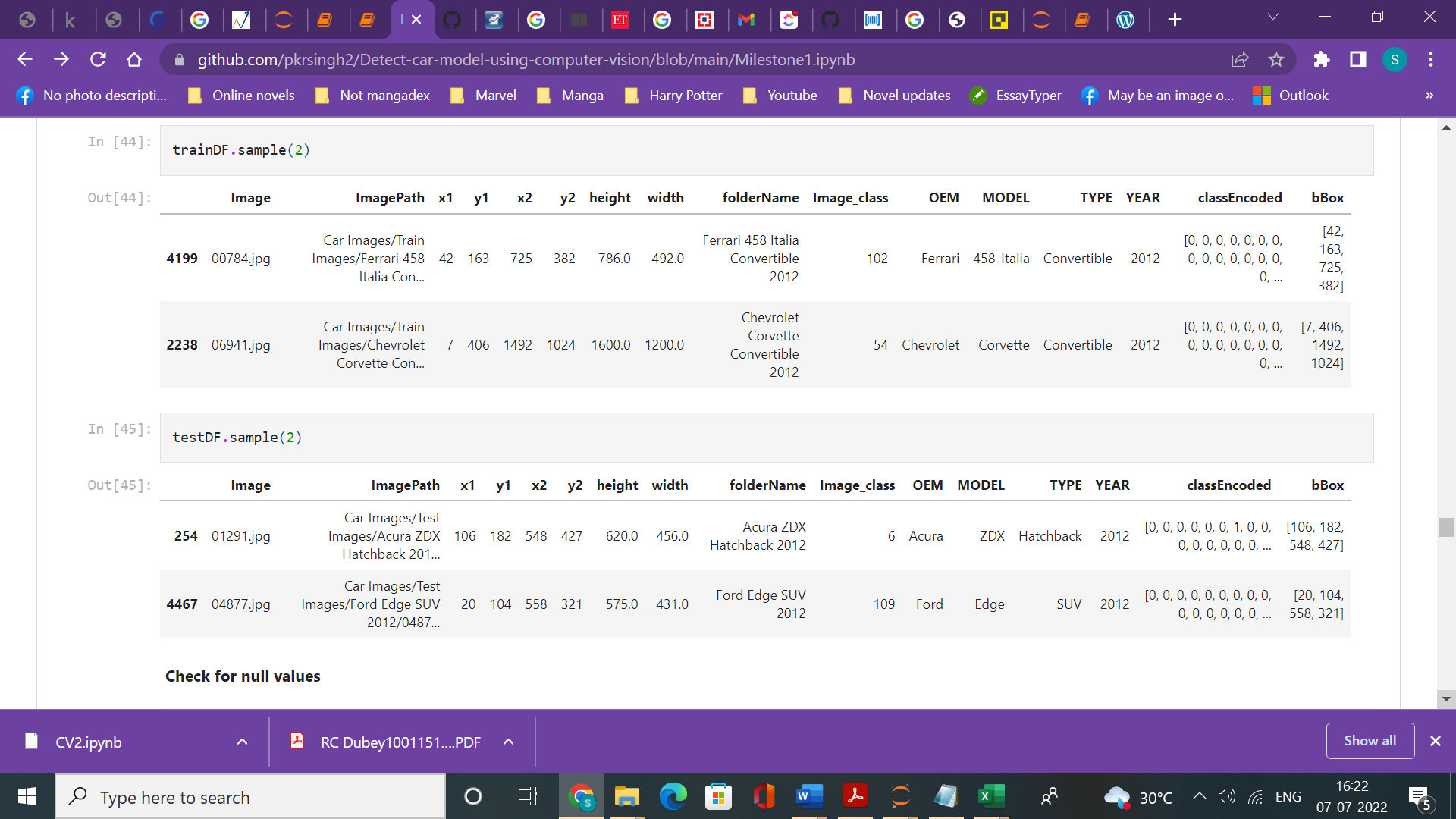
**Step 3: Validate data for any mismatch during merging**

After doing the cross merged and synced with "Train/Test Annotations.csv", "Car names and make.csv" and the images in the "Train/Test images folders", it is found to have no mismatch of information

There are now 22 columns in the data frame



**The data frame is now cleaned up, label encoding is done and a column to store the bounding box coordinates together is created**



Null values are checked and visualization of images with bounding boxes in

#### 

### **The data is now preprocessed with all the images of comparable size and augmented by bounding boxes for training through various models and algorithms**

### **Model Building**

### **The section details the steps for evaluating the appropriate model choices and preliminary results and insights from the initial runs**

### **Choosing an appropriate model algorithm**

### Coming up with a best model algorithm for the problem entailed an extensive exploratory study of various different models available and their appropriateness for being used for the problem in hand. We studied over 12 different computer vision models with respect to their mechanism, efficiency and appropriateness (annexure 1) The table below summarises the key features and pros and cons of the models for the problem in hand

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Model Net** | **Computation Requirement** |  |
| **1** | **VGG 16** |  |  |
| **2** | **Mobile Net** |  |  |
| **3** | **VGG 19** |  |  |
| **4** | **Mask RCNN** |  |  |
| **5** | **Single Shot Detector (SSD)** |  |  |
| **6** | **YOLO** |  |  |
| **7** | **SPPNET** |  |  |
| **8** | **Res NET** |  |  |
| **9** | **Inception Net** |  |  |
| **10** | **Faster RCNN** |  |  |
| **11** | **Google Net V2** |  |  |
| **12** | Ale Net |  |  |

### **Working with the initial model choices (Mobile Net)**

### We did an initial run using mobile net for classifcation. The screen shot below shares the configuration and the details of the model used. Mobile Net is a type of convolutional neural network designed for mobile and embedded vision applications. They are based on a streamlined architecture that uses depth wise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices.

### 

### Following are the advantages of using MobileNet over other state-of-the-art deep learning models.

### Reduced network size - 17MB.

### Reduced number of parameters - 4.2 million.

### Faster in performance and are useful for mobile applications.

### Small, low-latency convolutional neural network

### **Initial Model Training**

### Initial model training was performed using the mobilenet .

### Plain model training with existing training data and testing through test The results revealed data overfitting with training accuracy of about 50%

### 

### Training accuracy increased to 71% and test accuracy increased to almost 50%

### **Improving the Model Performance**

### **Data augmentation for get balanced data set**

### **Testing different model to come up best performing model**

### **Hyperparameter Tuning**

### **Future Work**

### **Evaluating performance of multiple models and algorithms to compare and come up with best performing model**

### **Improving model performance by Upscaling and downscaling the image data for balanced dataset, fine tuning parameters and hyperparameters of best algorithms**

### **Optimising and normalizing the image size to representative and efficient size**

### **Finalizing and integrating the UPI based support**

### **Overview of business case for the final model**

### **Advantages and limitations of the model**

### **Business Models**

### **On road car classification**

### **On road car surveillance**

### **City and town planning Designing and allocationg parking spaces**

### **Container capacity logistics to determine how many cars of different model can be shipped in the fied container size**

D

Anneture 1.

Evaluating pros and cons of various computer vision models

Annexure 2 Model -1 Configuration Mobile net ( Base data )

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_2 (InputLayer) [(None, 224, 224, 3)] 0

conv1 (Conv2D) (None, 112, 112, 32) 864

conv1\_bn (BatchNormalizatio (None, 112, 112, 32) 128

n)

conv1\_relu (ReLU) (None, 112, 112, 32) 0

conv\_dw\_1 (DepthwiseConv2D) (None, 112, 112, 32) 288

conv\_dw\_1\_bn (BatchNormaliz (None, 112, 112, 32) 128

ation)

conv\_dw\_1\_relu (ReLU) (None, 112, 112, 32) 0

conv\_pw\_1 (Conv2D) (None, 112, 112, 64) 2048

conv\_pw\_1\_bn (BatchNormaliz (None, 112, 112, 64) 256

ation)

conv\_pw\_1\_relu (ReLU) (None, 112, 112, 64) 0

conv\_pad\_2 (ZeroPadding2D) (None, 113, 113, 64) 0

conv\_dw\_2 (DepthwiseConv2D) (None, 56, 56, 64) 576

conv\_dw\_2\_bn (BatchNormaliz (None, 56, 56, 64) 256

ation)

conv\_dw\_2\_relu (ReLU) (None, 56, 56, 64) 0

conv\_pw\_2 (Conv2D) (None, 56, 56, 128) 8192

conv\_pw\_2\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_pw\_2\_relu (ReLU) (None, 56, 56, 128) 0

conv\_dw\_3 (DepthwiseConv2D) (None, 56, 56, 128) 1152

conv\_dw\_3\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_dw\_3\_relu (ReLU) (None, 56, 56, 128) 0

conv\_pw\_3 (Conv2D) (None, 56, 56, 128) 16384

conv\_pw\_3\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_pw\_3\_relu (ReLU) (None, 56, 56, 128) 0

conv\_pad\_4 (ZeroPadding2D) (None, 57, 57, 128) 0

conv\_dw\_4 (DepthwiseConv2D) (None, 28, 28, 128) 1152

conv\_dw\_4\_bn (BatchNormaliz (None, 28, 28, 128) 512

ation)

conv\_dw\_4\_relu (ReLU) (None, 28, 28, 128) 0

conv\_pw\_4 (Conv2D) (None, 28, 28, 256) 32768

conv\_pw\_4\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_pw\_4\_relu (ReLU) (None, 28, 28, 256) 0

conv\_dw\_5 (DepthwiseConv2D) (None, 28, 28, 256) 2304

conv\_dw\_5\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_dw\_5\_relu (ReLU) (None, 28, 28, 256) 0

conv\_pw\_5 (Conv2D) (None, 28, 28, 256) 65536

conv\_pw\_5\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_pw\_5\_relu (ReLU) (None, 28, 28, 256) 0

conv\_pad\_6 (ZeroPadding2D) (None, 29, 29, 256) 0

conv\_dw\_6 (DepthwiseConv2D) (None, 14, 14, 256) 2304

conv\_dw\_6\_bn (BatchNormaliz (None, 14, 14, 256) 1024

ation)

conv\_dw\_6\_relu (ReLU) (None, 14, 14, 256) 0

conv\_pw\_6 (Conv2D) (None, 14, 14, 512) 131072

conv\_pw\_6\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_6\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_7 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_7\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_7\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_7 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_7\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_7\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_8 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_8\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_8\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_8 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_8\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_8\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_9 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_9\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_9\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_9 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_9\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_9\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_10 (DepthwiseConv2D (None, 14, 14, 512) 4608

)

conv\_dw\_10\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_dw\_10\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_10 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_10\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_pw\_10\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_11 (DepthwiseConv2D (None, 14, 14, 512) 4608

)

conv\_dw\_11\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_dw\_11\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_11 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_11\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_pw\_11\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pad\_12 (ZeroPadding2D) (None, 15, 15, 512) 0

conv\_dw\_12 (DepthwiseConv2D (None, 7, 7, 512) 4608

)

conv\_dw\_12\_bn (BatchNormali (None, 7, 7, 512) 2048

zation)

conv\_dw\_12\_relu (ReLU) (None, 7, 7, 512) 0

conv\_pw\_12 (Conv2D) (None, 7, 7, 1024) 524288

conv\_pw\_12\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_pw\_12\_relu (ReLU) (None, 7, 7, 1024) 0

conv\_dw\_13 (DepthwiseConv2D (None, 7, 7, 1024) 9216

)

conv\_dw\_13\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_dw\_13\_relu (ReLU) (None, 7, 7, 1024) 0

conv\_pw\_13 (Conv2D) (None, 7, 7, 1024) 1048576

conv\_pw\_13\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_pw\_13\_relu (ReLU) (None, 7, 7, 1024) 0

global\_average\_pooling2d\_1 (None, 1024) 0

(GlobalAveragePooling2D)

dense\_3 (Dense) (None, 1024) 1049600

dense\_4 (Dense) (None, 512) 524800

dense\_5 (Dense) (None, 196) 100548

=================================================================

Total params: 4,903,812

Trainable params: 1,674,948

Non-trainable params: 3,228,864

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_