CAPSTONE PROJECT - FINAL REPORT

Submitted by - AIML Batch-B Jun, 23

OBJECT DETECTION - CAR

BOUNDING BOX REGRESSION   AND MULTICLASS CLASSIFICATION

OF STANFORD CAR IMAGES

Mentor - **Mr. Jyant Mahara**

Members - **Radhamadhavi, Sunil, Srikanth, Sneha**

Contents

SECTION 1: INTRODUCTION

Summary of the Problem statement

Project Objective

The aim is to undertake a multi-faceted project that demonstrates our understanding and mastery of the key conceptual and technological aspects of Deep Learning. Project also facilitates understanding the current scenarios in deep learning, the practicalities and the trade-offs that need to be made when solving a problem in real life.

The Stanford 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. The Objective is to design a DL based car identification model for the Stanford Car Images. The problem statement is a Multi-class object detection which involves

(1) detection of where an object is in an input image [Bounding box]
(2) prediction of what the detected object is [Class].

Dataset Description:

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
Car Make: consists of list of car make, name and year list

Data and Findings

Data

Image folders 'Train images' and 'Test images' contain the images of the dataset carefully placed in the appropriate folder named after the car make, model and the year. "Car names and make.csv" file has the list of the cars in the folders.

Train and Test Annotation csv files contain the mapping between the image (file name in this case) and the output labels (class name and bounding box coordinates).

Findings:

a) The bounding box coordinates in Train and Test annotation csv files are not clearly labelled. It could be either in (xmin, ymin, xmax, ymax) format or (x, y, width, height) format. Drawing bounding boxes using both the formats allowed us to identify that they were indeed given in (xmin, ymin, xmax, ymax) format.

b) The number of car names in "Car names and make.csv" file match the number of unique make of cars in the training annotations file. Verifying this ensured none of the folders were missed during training.

c) The number of images in each class are evenly balanced.
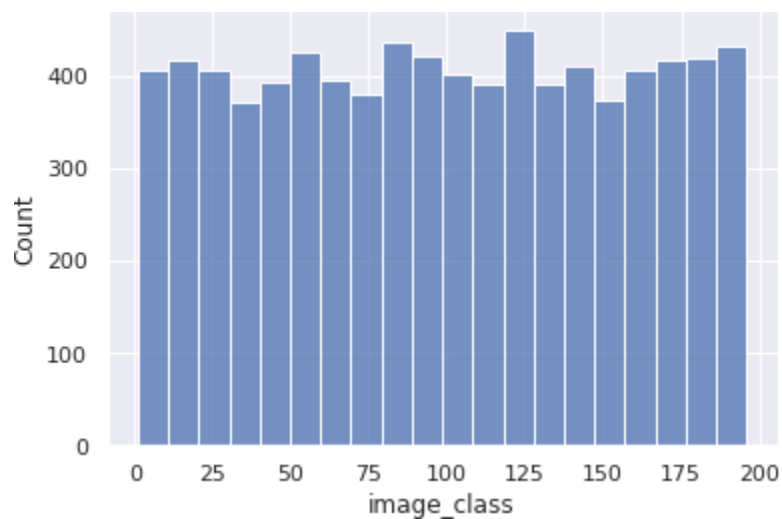
SOLUTION OVERVIEW

The problem statement states that it is in the Automotive domain. Surveillance problem and the context states that computer vision can be used to automate supervision and triggering events for images of interest.

For this problem statement we decided to use transfer learning and use a pre-built model in tensorflow. Few layers in pre-built models can be trained to get good accuracy and best prediction for the bounding box. Transfer learning is adaptable, allowing pre-trained models to be used directly as feature extraction preprocessing or integrated into completely new models. We have evaluated a few pre-trained models - MobileNet, ResNet50 and efficient net as part of

the experiment. Among these evaluations we have got the best result for an efficient net - efficientnet-b7.

As suggested, GUI is built to demonstrate the automation and manageability of the overall process. The sections that follow elaborates on the process, implementation and our learnings based out of this project.

SECTION 2: PROCESS

Overview of the process

Salient features of the data

Loading Annotations file in a pandas Dataframe object. Each row in Annotations file consists of consists of six elements:
1. Image Filename
2. Starting x-coordinate
3. Starting y-coordinate
4. Ending x-coordinate
5. Ending y-coordinate

6. Class label number

Image file name points to the Images in the respective folder. Images in the train/test images come in all sizes and dimensions. Part of the pre-processing includes resizing the image (to 224X224) and scaling up the bounding box coordinates to prepare the same for data modelling.

Overview of the Placement of the cars in the image:
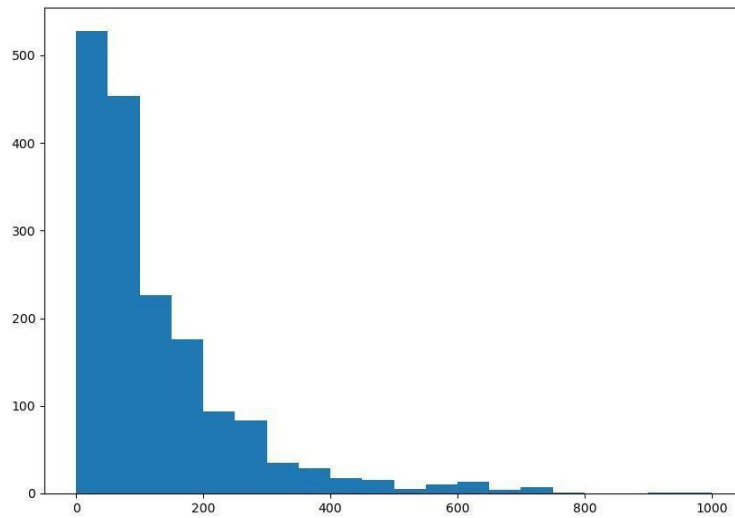
Fig1. Car Placement in the image

Overview of Original Dimension of the images from Train folder

DIM1:

+ Code   + Text

```
[17] import seaborn as sns
     print("Dimension Analysis:\n")
     val_count = dim_analysis['dim1'].value_counts()
     print(val_count)
```

```
Dimension Analysis:

640     1286
1024     679
300      482
500      411
259      404
       ...
3648       1
526        1
527        1
1555       1
1542       1
Name: dim1, Length: 654, dtype: int64
```

```
[18]  sns.histplot(dim_analysis['dim1'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb057a80910>
```

+ Code   + Text

```
[15] print(val_count)
```

```
Dimension Analysis:

480     1006
768      478
194      414
360      307
225      210
       ...
2850       1
799        1
296        1
794        1
2048       1
Name: dim2, Length: 829, dtype: int64
```

```
sns.histplot(dim_analysis['dim2'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe55c101b50>
```


✓ 0s   completed at 7:40 PM

Fig2. Dim1 of Train images

DIM2

Fig3. Dim2 of Train images

Data Pre-processing

For Preprocessing images, below mentioned simple 3 steps are followed.

1. Reading images from images folder
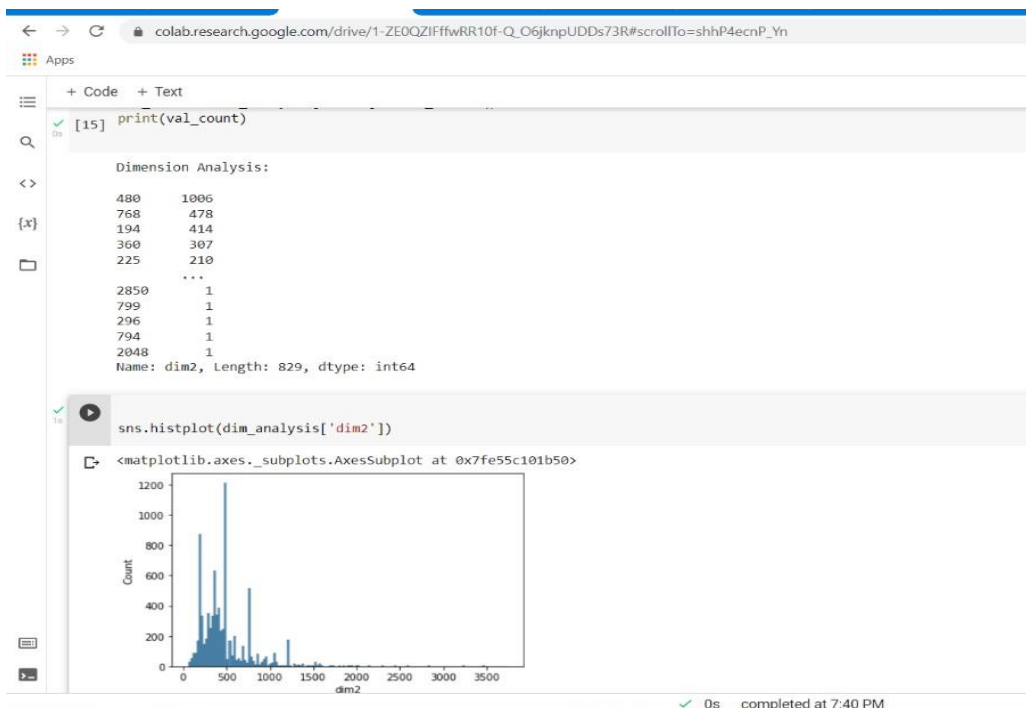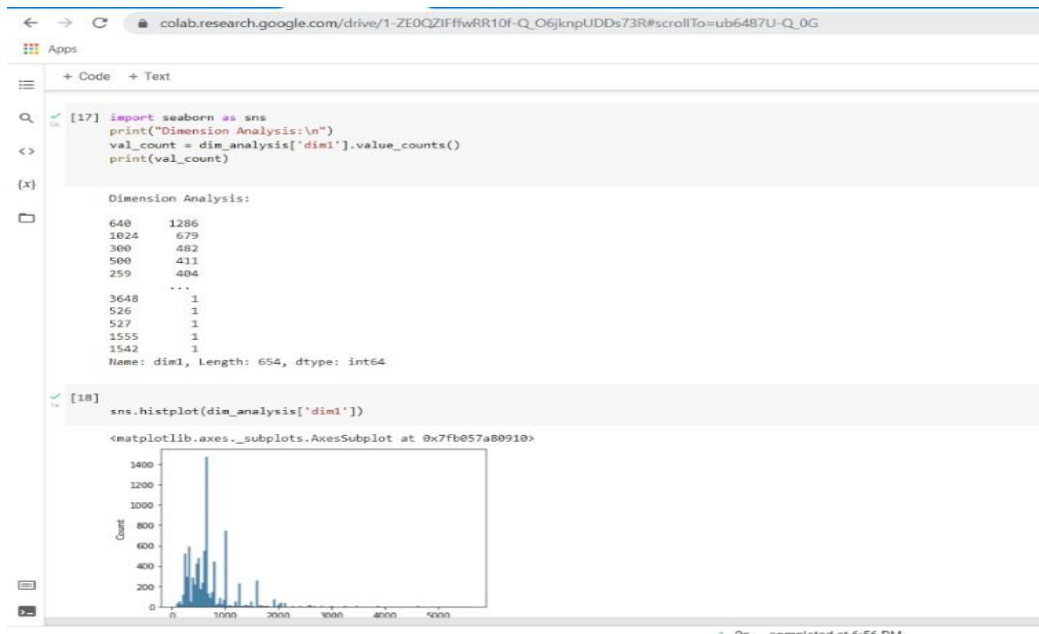2. Reading annotation file
3. Extracting image height and width
4. Merging image annotation dataset with data produced in above 2 steps.

Creating dataset from images folder

The images subdirectory contains all images in our dataset, with a corresponding subdirectory for the name of the label. We start with loading the Car Name and Class Name into a pandas dictionary object. Looping over our CSV annotation files, we grab all rows in the file and proceed to loop over each of them and retrieve the required image information.

Finally, we load the image from disk in Keras/TensorFlow format and preprocess it with a resizing step which forces the image to 224×224 pixels for input for the model. Next step is to One-hot encode our labels using LabelBinarizer.

Input – 224x224 image matrix

Output – Class label, 4 bounding box coordinates

The Algorithms

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

Two main parts to a CNN architecture supports and helps in solving the problem in hand and ant image related problems in general.

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
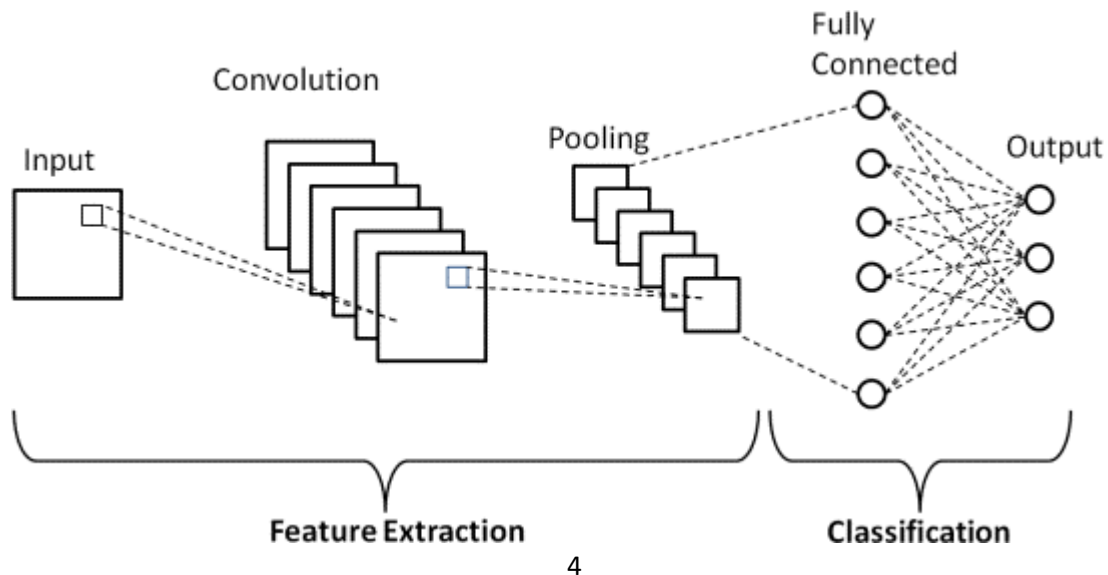
Fig4. CNN architecture overview

In simple mathematical terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

Object Classification in this case is multi-class classification. And Bounding box regression is used for Object localization Bounding-box regression is a popular technique in object detection algorithm used to predict target objects' location using rectangular bounding boxes. It aims to refine the location of a predicted bounding box.

The aim was to define and model an algorithm that would do object classification and object localization in its best accuracy. Various of the following algorithms were considered and executed before Efficientnet-b7 was pickled for the purpose. Few of the pre-trained models - MobileNet, ResNet50 and EfficientNet-B5, YOLO and TFOD as part of the experiment.

GUI

UI interface demonstrates the Data loading, data preprocessing, automates the data modelling steps and enables to test and predict the classification and localization on the pickled data model. Various options like Tkinter, Flask and Ngrok were adapted and experimented for the purpose.

Flask is a Python eb framework built with a small core and easy-to-extend philosophy. To implement, one imports the Flask class. An instance of this class will be our WSGI application. WSGI is the Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.

Ngrok is a tool that allows you to expose a web server running on your local machine to the internet. It provides a real-time web UI where one can introspect all HTTP traffic running over your tunnels. Replay any request against the tunnel with one click.

The **tkinter** package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. The Tk class is instantiated without arguments. This creates a toplevel widget of Tk which usually is the main window of an application. Each instance has its own associated Tcl interpreter.

Streamlit is an open-source app framework for Machine Learning and Data Science teams. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

Flask and docker; Heroku; AWS deployment were the other options considered while working on GUI solution

SECTION 3: SOLUTION

Google Colab was extensively used for data preprocessing and modelling. It was chosen for the ability to handle huge datawork and extended GPU facilities. The complete solution was broken down into three steps/Milestone.
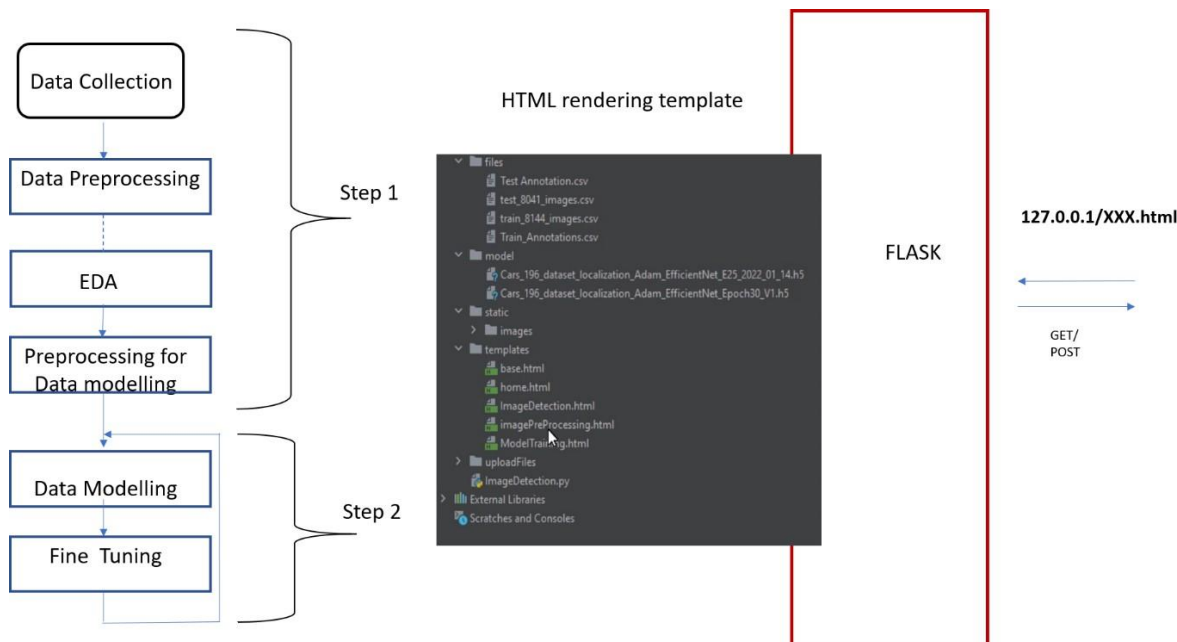
Fig5. Overall Solution Overview

Step 1 as illustrated in Data collection, Data preprocessing and EDA part.

Step 2 involves the iterative process of model building, model evaluation and fine tuning.

Step 3 involves the GUI that simulates step 1 and step2
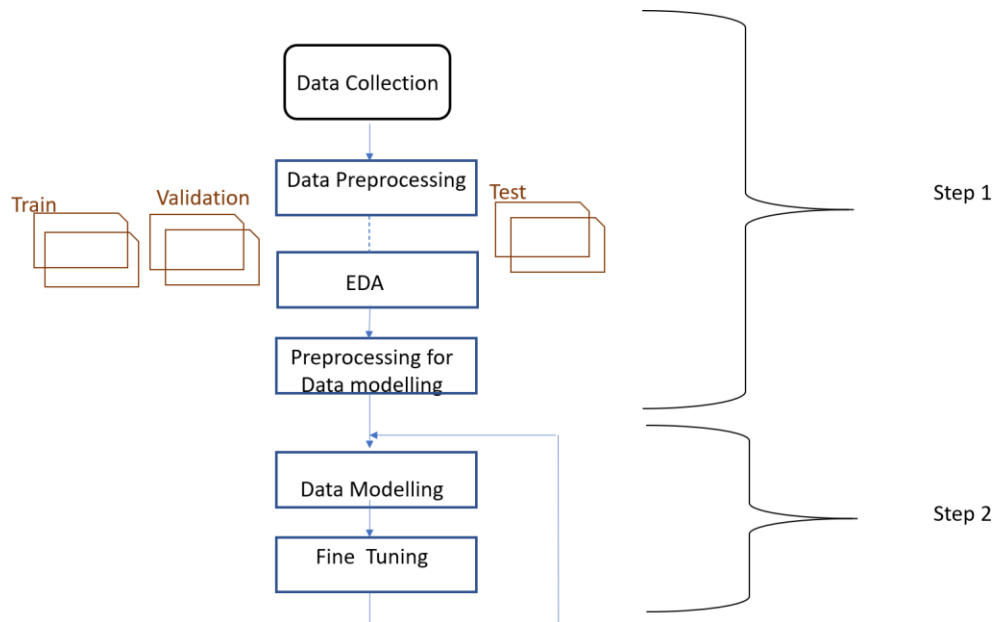
Fig6. Data Modeling Solution Overview

STEP1

Data loading

The Car Name and Class Name is loaded into a pandas dictionary object. Looping over our CSV annotation files, we grab all rows in the file and proceed to loop over each of them.

Inside our loop, we unpack the comma-delimited row giving us our filename, (x, y)-coordinates, and class label for the particular line in the CSV. We again loop through the annotations data frame and update new columns for Class Name and build Image path name. Using the imagePath derived from our config, class label, and filename, we load the image and extract its spatial dimensions. We then scale the bounding box coordinates relative to the original image 's dimensions to the range [0, 1] — this scaling serves as our preprocessing for the bounding box data.
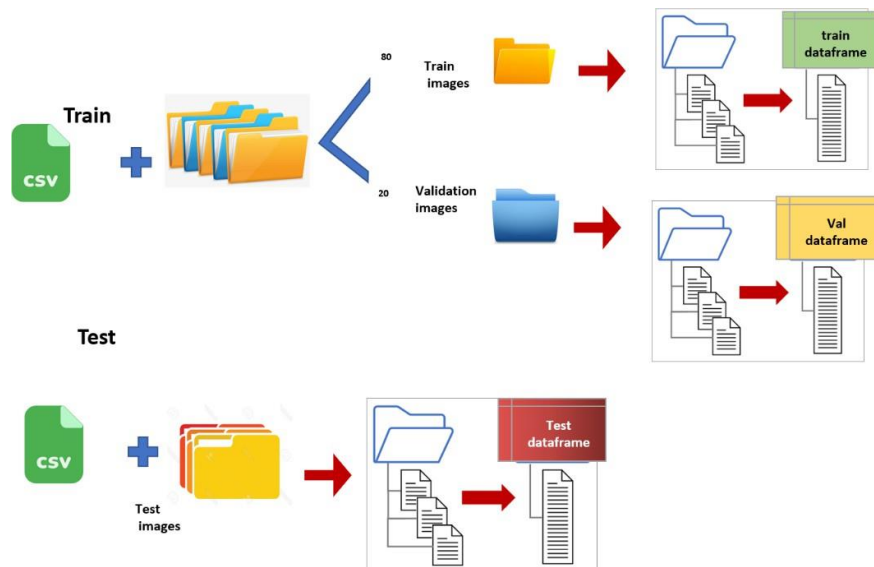
Fig7. Date loading Overview

Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model.

In this project, it involves extracting the image information and preprocess it to be suitable in Keras/TensorFlow format and scaling to suitable dimension.

Input – 224x224 image matrix
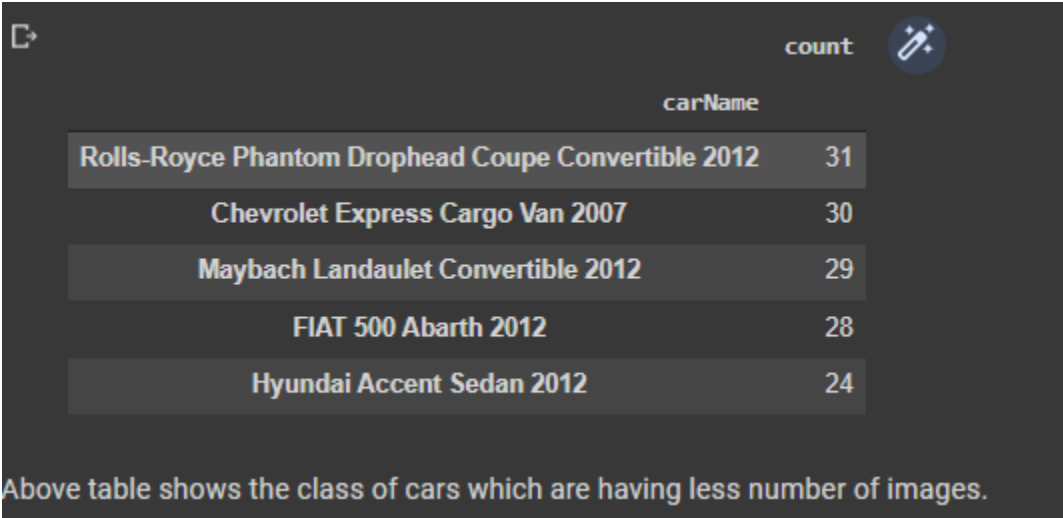
And for the Output – Class label, 4 bounding box coordinates

The output label is One-hot encoded using LabelBinarizer. The bounding box coordinates are scaled relative to the original image 's dimensions w.r.t the 224*224 dimension.

Exploratory data analysis

Class having minimum number of images in train dataset

Below mentioned picture shows 5 classes which are having less number of images in the Train dataset.

carName	count
Rolls-Royce Phantom Drophead Coupe Convertible 2012	31
Chevrolet Express Cargo Van 2007	30
Maybach Landaulet Convertible 2012	29
FIAT 500 Abarth 2012	28
Hyundai Accent Sedan 2012	24

Above table shows the class of cars which are having less number of images.

Fig8. Table having min number of images in Train dataset

Class having minimum number of images in test dataset

Below mentioned picture shows 5 classes which are having less number of images in the Test dataset.

carName	count
Rolls-Royce Phantom Drophead Coupe Convertible 2012	30
Chevrolet Express Cargo Van 2007	29
Maybach Landaulet Convertible 2012	29
FIAT 500 Abarth 2012	27
Hyundai Accent Sedan 2012	24

Fig9. Table having min number of images in Test dataset

Class having maximum number of images in test dataset

Below mentioned picture shows 5 classes which are having more number of images in the Test dataset.

carName	count
GMC Savana Van 2012	68
Chrysler 300 SRT-8 2010	48
Mercedes-Benz 300-Class Convertible 1993	48
Mitsubishi Lancer Sedan 2012	47
Audi S6 Sedan 2011	46

Fig10. Table having max number of images in Train dataset

Number of car images based on Year

Below mentioned picture depicts the number of car images based on car model year. There are more images of cars which are of model 2012 and very less images belonging to model 1996.
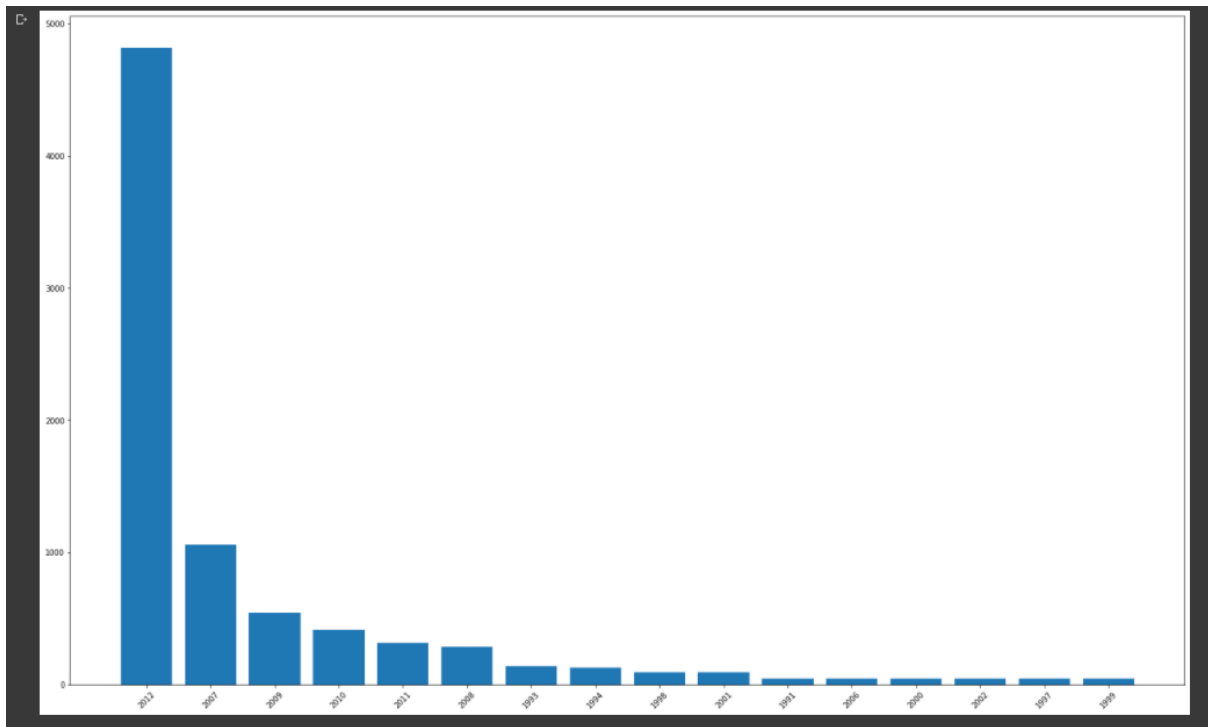
Fig11. Cars Vs year of make

Class distribution

Almost all classes of cars images have an equal number of images which is of count around 40. There is a class named "GMC savana van 2012" which has 60 images and "Hyundai Accent sedan 2012" which has around 25 images. In below mentioned pictures we can view the class distribution.

Fig12, 13, 14. Car count histogram (of 196 classes)

Maximum Size of the image available in dataset

Image with name 05945.jpg which belongs to class "Chevrolet Sonic Sedan 2012" is having maximum height and width. The dataset has uneven distribution of the image size and hence preprocessing is required before running the model on data. While preprocessing we have to define a fixed image size and the tensorflow preprocessing function was applied on each images before training the model.

	carName	imageName	Height	Width
2779	Chevrolet Sonic Sedan 2012	05945.jpg	3744	5616

Fig15. Maximum Size of the image available in dataset

STEP2

Design

Owing to the size of the data to be handled and the GPU facility, Google colab was predominately used for data preprocessing and data modeling phase.

By understanding this problem, we concluded that the problem can be solved by using deep learning techniques. In Deep learning techniques, we can investigate particular sections of CNNs. Because of its great accuracy, CNNs are employed for image classification, image localization, image detection, etc. The CNN uses a hierarchical model that builds a network, similar to a funnel, and then outputs a fully connected layer in which all neurons are connected to each other, and the output is processed. The benefit of using CNNs is their ability to develop an internal representation of an image by looking at only a subset of pixels in the images.

We cannot use a Dense neural network for computer vision tasks in deep learning. The fundamental difference between the Convolutional and Dense layers is that the Convolutional layer requires fewer parameters because the input values are forced to share the parameters. The Dense Layer employs a linear operation, which means that the function generates each output based on each input. An output of the convolution layers is formed by just a small size of inputs which depends on the filter's size and the weights are shared for all the pixels

Deploy

For this problem statement we have decided to use transfer learning and use a pre-built model in tensorflow. Few layers in pre-built models can be trained to get good accuracy and best prediction for the bounding box. Transfer learning is adaptable, allowing pre-trained models to be used directly as feature extraction preprocessing or integrated into completely new models. We have evaluated a few pre-trained models - MobileNet, ResNet50 and efficient net as part of the experiment. Among these evaluations we have got the best result for an efficient net - efficientnet-b7

```
model = tf.keras.applications.efficientnet.EfficientNetB7(include_top=False,
                                        input_shape=(224, 224, 3),
                                        weights='imagenet')
```

Few layers of pre-trained model were unfreezed for more training. The output layer of pretrained model was flattened and tuned for label classification training (for 196 classes) using softmax activation and

bounding box regression training (for 4 outputs) using sigmoid activation dense layer. This non-sequential model output was then trained for categorical crossentropy and userdefined IOU metrics respectively over the Adam optimizer.

Model Pickling

Final model was pickled in h5 format, this file was later used to load the model (with compile false and custom_objects properly defined) to be used for predict and draw in step3.

STEP3

GUI

In flask, we use a render_template method to render any HTML page on our browser. Templates are files that contain static data as well as placeholders for dynamic data. A template is rendered with specific data to produce a final document. Flask uses the Jinja template library to render templates.

Each page in the application will have the same basic layout around a different body. Instead of writing the entire HTML structure in each template, each template will *extend* a base template and override specific sections.

Fig 16. Template directory
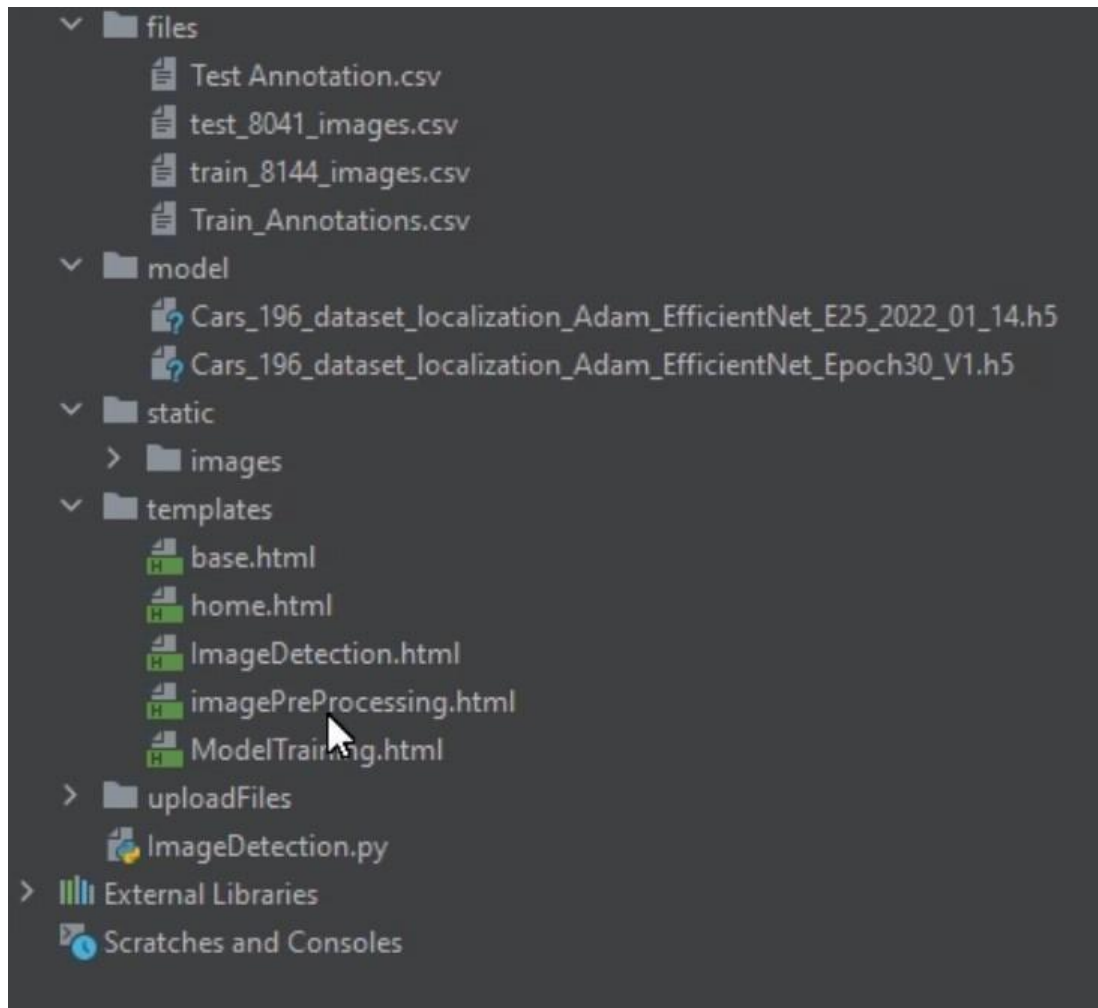
We are importing **render_template** function provided by the Flask and then rendering our HTML template in the home route. And run the using "flask run" command.

App routing is used to map the specific URL with the associated function that is intended to perform some task.

Note: For the implementation ease, step1 and step 2 are the planned minuscule set of the complete Dataset.

base.html

base.html contains the CSS formatting, basic navigation bar links to the three steps of operation that is automated. It is the basic skin for all the three different html.

Capstone Batch Jan 21-B   Image preprocessing and EDA   Model training   Image detection
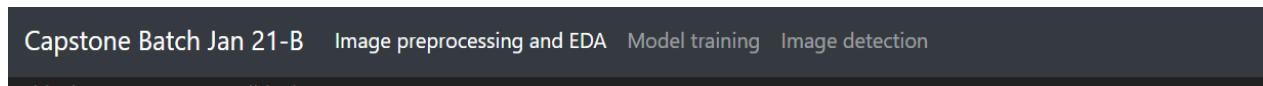
Fig17. Base.html

ImageProcessing.html

The annotation csv file and image folder are POSTed and the image processing is done rendering you the details about the data followed specific EDA outputs. EDA output images gets saved in the static/images directory and rendered as template to the html.

EDA and Pre processing input files

Upload annotations file

Upload File: Choose File No file chosen   images directory [   ]

Upload

Fig18. Imageprocessing.html format

ModelTraining.html

Model training is done on subset of the original dataset. User has the ability to the dictate the number of epochs upon which the model training is initiated and the Accuracy details are plotted on the completion of the model training.

Press below button to start training

number of epochs [   ] Train Model

Fig19. Modeltraining.html form

ImageDetection.html

Image detection page enables the user to submit the test images. The test images get predicted on the pickled model. The pickled model is saved EfficientNet-B7 model of the actual training that was originally done in Google colab.

Image Detection

Upload a new image for which image detection needs to be run

Choose File | No file chosen

Submit

Fig 20. Imagedetection.html format

SECTION 4: MODEL EVALUATION

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. The various models were tried and evaluated before the best model was finalized. The methods and benchmarks for the evaluation has been detailed below.

Classification Parameters

Categorical cross entropy is a loss function that is used in this multi-class classification task. It is designed to quantify the difference between two probability distributions. Mathematically,

$$Loss = -\sum_{i=1} t_{yi} \cdot log \hat{y}^i$$

where
$\hat{y}_{i}y^i$ is the ii-th scalar value in the model output,
y_iyi is the corresponding target value, and it is summed over the output size is the number of scalar values in the model output.

A **Classification report** is used to measure the quality of predictions from a classification algorithm. It is used to evaluate the model. The report shows the main classification metrics precision, recall and f1-score on a per-class basis.

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$accuracy = correct predictions \,/\, all predictions$$

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$precision = true positives/(true positives + false positives)$$

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$recall = true positives/(true positives + false negatives)$$

The **F1 score** is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.

$$F1\ Score = 2*(Recall * Precision) / (Recall + Precision)$$

Object Detection Parameter

Mean average precision (mAP) is used to determine the accuracy of a set of object detections from a model when compared to ground-truth object annotations of a dataset.

Intersection over Union (IoU) is used when calculating mAP. It is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box.
> an IoU of 0 means that there is no overlap between the boxes
> an IoU of 1 means that the union of the boxes is the same as their overlap indicating that they are completely overlapping

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

Fig 21. IOU explained

Model Building

The following are the various models compiled and evaluated on the based on the parameters listed above.

MobileNet

MobileNets are built on a simplified design that builds light weight deep neural networks using depth-wise separable convolutions. Two simple global hyper-parameters are introduced that efficiently trade off latency and accuracy. These hyper-parameters help the model builder to select the appropriate model size for their application based on the problem constraints. We are using a specific version of mobile net model MobileNetV2. MobileNetV2 is very similar to the original MobileNet, except

that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance.

The network is a fairly simple network with less number of weights to be trained. We are retraining the model right from input layer and we have added dense layer and dropout layer along with batch normalization as final layer for classification and regression (calculation of bounding box).

We are using categorical_crossentropy for classification and mse for regression. We calculate the overall loss of the model by adding regression loss and classification loss. For calculating accuracy of the bounding box on the images, we have defined a function to calculate IOU. This Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

Below mentioned are the weight which are to be learnt for mobile net model

Total params: 4,988,104
Trainable params: 4,953,352
Non-trainable params: 34,752

Resnet50

ResNet50 is a ResNet variation of 48 Convolution layers, 1 MaxPool layer, and 1 Average Pool layer. There are 3.8 x 109 floating point operations in it. It's a popular ResNet model. ResNet is a technique for dealing with the vanishing gradient problem in deep CNNs. They work by bypassing some layers, believing that very deep networks should not have a higher training error than shallower networks.

Here we have retrained the entire model, this model has 28 Million trainable parameters. The Last layer of the model has GlobalAveragePooling2D, 2 Dense layer, 1 dropout layer and a Batch normalization layer. For classification we have used softmax activation function with 196 classes and for regression we have used sigmoid activation function for 4 coordinates of the bounding box.

We are using categorical_crossentropy for classification and MSE for regression. We calculate the overall loss of the model by adding regression loss and classification loss. For calculating accuracy of the bounding box on the images, we have defined a function to calculate IOU. This Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

Below mentioned are the weight which are to be learnt for mobile net model

Total params: 28,937,800
Trainable params: 28,883,656
Non-trainable params: 54,144

Efficientnet-b5

Efficientnet-b5 model with ImageNet weights is used for developing network for classification and regression tasks for the given problem statement. The efficientnet-b5 model is one of the EfficientNet models designed to perform image classification. All the EfficientNet models have been pre trained on the ImageNet image database.

Efficientnet-b5 has 576 total layers, from layer number 257 we have made it trainable. This model has 28 Million trainable parameters. The Last layer of the model has GlobalAveragePooling2D, 1 Dense layer and a Batch normalization layer. For classification we have used softmax activation function with 196 classes and for regression we have used sigmoid activation function for 4 coordinates of the bounding box.

We are using categorical_crossentropy for classification and MSE for regression. We calculate the overall loss of the model by adding regression loss and classification loss. For calculating accuracy of the bounding box on the images, we have defined a function to calculate IOU. This Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

Number of weights after final model building is as mentioned below.

Total params: 33,127,871
Trainable params: 31,496,968
Non-trainable params: 1,630,903

Efficientnet-b7

Efficientnet-b7 model with ImageNet weights is used for developing network for classification and regression tasks for the given problem statement. The efficientnet-b7 model is one of the EfficientNet models designed to perform image classification. All the EfficientNet models have been pre trained on the ImageNet image database.

Efficientnet-b7 has 813 total layers, from layer number 351 we have made it trainable. This model has 64 Million trainable parameters. The Last layer of the model has GlobalAveragePooling2D, 1 Dense layer and a Batch normalization layer. For classification we have used softmax activation function with 196 classes and for regression we have used sigmoid activation function for 4 coordinates of the bounding box.

We are using categorical_crossentropy for classification and MSE for regression. We calculate the overall loss of the model by adding regression loss and classification loss. For calculating accuracy of the bounding box on the images, we have defined a function to calculate IOU. This Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

Number of weights after final model building is as mentioned below.

Total params: 71,176,287
Trainable params: 67,594,112
Non-trainable params: 3,582,175

SECTION 5: BENCHMARK

Benchmarking is used to measure performance using a specific indicator resulting in a metric that is then compared to others. In other words, it is the comparison of a given model's inputs and outputs to estimates from alternative internal or external data or models

Test Results Comparison

MobileNet being a streamlined architecture, has shown good accuracy for train and validation set of data but performs poorly on unseen test dataset.

ResNet50 performs poorly on unseen test dataset.

EfficientNet-B5 performs well on unseen test dataset

EfficientNet-B7 performs well on unseen test dataset yielding better accuracy numbers.

Test results of the above-mentioned models in the table below.

Model	loss	Class_op_loss	reg_op_loss	class_op_accuracy	reg_op_IoU
MobileNet	1.143170595	1.140172958	0.002998107346	0.7411249876	0.8120642304
ResNet50	4.882508278	4.872550488	0.009955173358	0.06724999845	0.7253609896
EfficientNet-B5	1.271269798	1.268125296	0.003145831171	0.7512500286	0.8126088381
EfficientNet-B7	1.153328776	1.150439143	0.002889547963	0.7966250181	0.8201269507

Fig22. Result comparison

EfficientNet-B7 model Numbers

The finalized model EfficientNet-B7 is trained in 2 phases, 1st phase with 10 epochs and batch size as 64 and in 2nd phase 15 more epochs with batch size as 16.
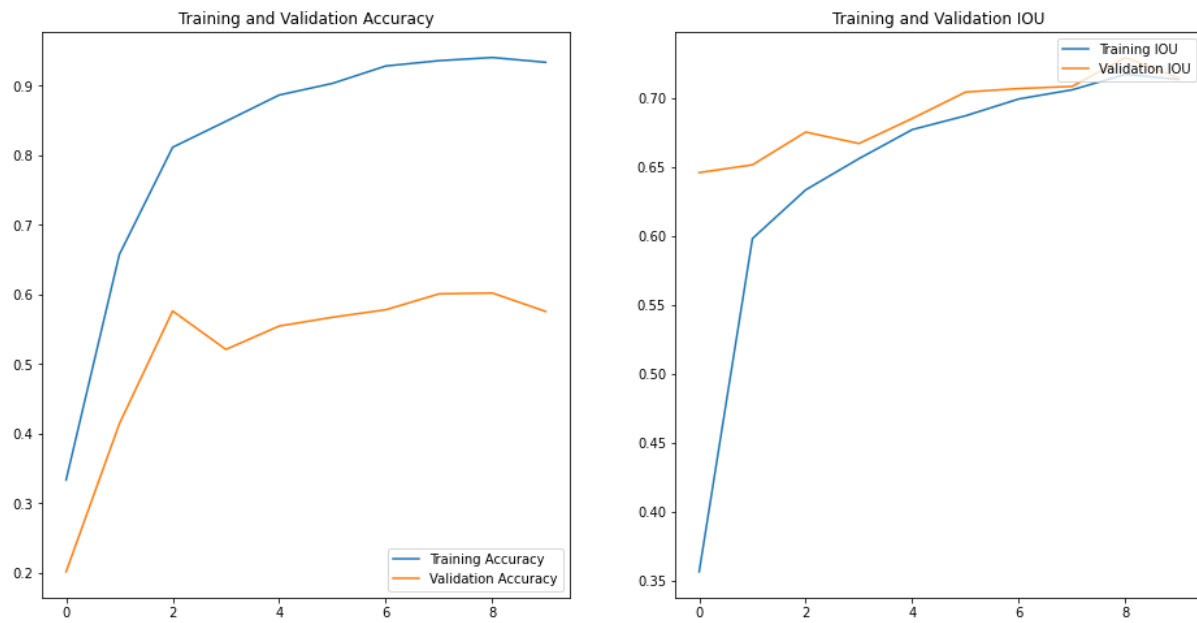
Fig22. First Phase results

Above figure shows the training vs validation accuracy and Training vs validation IOU in the first phase.
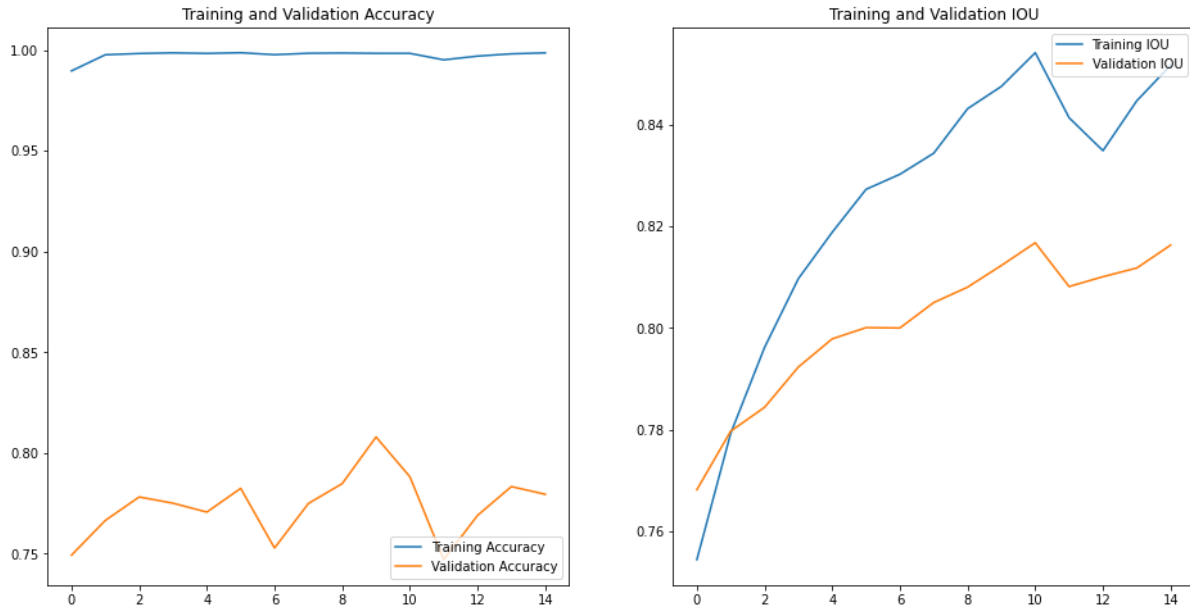
Fig23. Second Phase results

Above figure shows the training vs validation accuracy and Training vs validation IOU in the second phase.

The training vs validation accuracy in the first phase is having a gap between them but with the second phase the validation accuracy has improved. Although the observed gap is more, for a check on a few thousand sample data, the classification seems to be acceptable. The IOU of the model has performed well and compared to resnet50 and MobileNet, the bounding boxes on the sample data has shown improvement. Below mentioned are a few samples of the dataset.

Fig24. Output Image 1

Real Label: Dodge Ram Pickup 3500 Crew Cab 2010

Predicted Label: Dodge Ram Pickup 3500 Crew Cab 2010

Fig25. Output Image 2

Real Label: BMW ActiveHybrid 5 Sedan 2012

Predicted Label: BMW ActiveHybrid 5 Sedan 2012

Based on the performance, the EfficientNet-B7 model is finally chosen, which is having the best performance among all the ones we have seen so far.

Classification report

Below mentioned table classification report for EfficientNet-B7 model. Overall Accuracy is 79%, Precision is 78%, recall is 78% and f1-Score is 78%. As observed in the classification report, for a few classes the number is low and the same is observed in test samples where the model is showing misclassification. The number of samples used for producing reports is 8041. In EfficientNet, the authors propose a new Scaling method called Compound Scaling.

Classification report for EfficientNet-B7

	precision	recall	f1-score	support
Acura Integra Type R 2001	0.89	0.95	0.92	44
Acura RL Sedan 2012	0.92	0.8	0.85	44
Acura TL Sedan 2012	0.59	0.62	0.61	32
Acura TL Type-S 2008	0.81	0.88	0.84	43
Acura TSX Sedan 2012	0.78	0.83	0.8	42
Acura ZDX Hatchback 2012	0.88	0.7	0.78	40
AM General Hummer SUV 2000	0.68	0.72	0.7	39
Aston Martin V8 Vantage Convertible 2012	0.63	0.6	0.61	45
Aston Martin V8 Vantage Coupe 2012	0.63	0.54	0.58	41
Aston Martin Virage Convertible 2012	0.68	0.64	0.66	33
Audi 100 Sedan 1994	0.74	0.82	0.78	38
Audi 100 Wagon 1994	0.68	0.57	0.62	40
Aston Martin Virage Coupe 2012	0.64	0.83	0.72	42
Audi A5 Coupe 2012	0.57	0.88	0.69	41
Audi S4 Sedan 2007	0.68	0.91	0.78	43
Audi RS 4 Convertible 2008	0.79	0.92	0.85	36
Audi R8 Coupe 2012	0.87	0.73	0.8	45
Audi S5 Convertible 2012	0.66	0.59	0.62	39
Audi S4 Sedan 2012	0.77	0.55	0.64	42

Audi S5 Coupe 2012	0.49	0.43	0.46	42
Audi TT Hatchback 2011	0.84	0.8	0.82	46
Audi TT RS Coupe 2012	0.42	0.62	0.51	40
Audi TTS Coupe 2012	0.7	0.59	0.64	39
Bentley Continental Flying Spur Sedan 2007	0.53	0.45	0.49	42
Audi V8 Sedan 1994	0.68	0.63	0.65	43
Bentley Arnage Sedan 2009	0.77	0.69	0.73	35
Audi S6 Sedan 2011	0.71	0.83	0.76	41
Bentley Continental GT Coupe 2007	0.69	0.79	0.73	42
Bentley Continental Supersports Conv. Convertible 2012	0.82	0.8	0.81	41
Bentley Continental GT Coupe 2012	0.72	0.64	0.67	44
BMW 1 Series Convertible 2012	0.85	0.68	0.75	34
BMW 1 Series Coupe 2012	0.73	0.82	0.77	44
Bentley Mulsanne Sedan 2011	0.8	0.68	0.74	41
BMW 3 Series Sedan 2012	0.61	0.49	0.54	41
BMW 3 Series Wagon 2012	0.69	0.89	0.78	38
BMW 6 Series Convertible 2007	0.86	0.78	0.82	41
BMW M3 Coupe 2012	0.71	0.76	0.74	42
BMW M5 Sedan 2010	0.74	0.88	0.8	40
BMW M6 Convertible 2010	0.76	0.87	0.81	39
BMW ActiveHybrid 5 Sedan 2012	0.71	0.82	0.76	44
BMW X5 SUV 2007	0.58	0.72	0.64	46
BMW X3 SUV 2012	0.78	0.53	0.63	34

BMW X6 SUV 2012	0.67	0.67	0.67	36
Bugatti Veyron 16.4 Coupe 2009	0.79	0.66	0.72	35
Bugatti Veyron 16.4 Convertible 2009	0.58	0.78	0.67	32
BMW Z4 Convertible 2012	0.8	0.74	0.77	43
Buick Enclave SUV 2012	0.71	0.86	0.77	42
Buick Regal GS 2012	0.9	0.86	0.88	42
Buick Rainier SUV 2007	0.81	0.74	0.78	35
Cadillac Escalade EXT Crew Cab 2007	0.72	0.84	0.77	37
Buick Verano Sedan 2012	0.85	0.93	0.89	43
Cadillac CTS-V Sedan 2012	0.93	0.86	0.89	44
Cadillac SRX SUV 2012	0.93	0.95	0.94	41
Chevrolet Camaro Convertible 2012	0.74	0.78	0.76	45
Chevrolet Cobalt SS 2010	0.74	0.77	0.76	44
Chevrolet Avalanche Crew Cab 2012	0.94	0.76	0.84	41
Chevrolet Corvette Ron Fellows Edition Z06 2007	0.66	0.69	0.68	39
Chevrolet Corvette ZR1 2012	0.8	0.97	0.88	37
Chevrolet Corvette Convertible 2012	0.94	0.74	0.83	46
Chevrolet Express Cargo Van 2007	0.36	0.17	0.23	29
Chevrolet HHR SS 2010	0.4	0.66	0.5	35
Chevrolet Express Van 2007	0.85	0.97	0.91	36
Chevrolet Malibu Hybrid Sedan 2010	0.67	0.74	0.7	43
Chevrolet Malibu Sedan 2007	0.81	0.79	0.8	38
Chevrolet Monte Carlo Coupe 2007	0.78	0.8	0.79	44
Chevrolet Impala Sedan 2007	0.87	0.76	0.81	45

Chevrolet Silverado 1500 Classic Extended Cab 2007	0.79	0.81	0.8	42
Chevrolet Silverado 1500 Hybrid Crew Cab 2012	0.73	0.84	0.78	43
Chevrolet Silverado 1500 Extended Cab 2012	0.71	0.5	0.59	40
Chevrolet Silverado 2500HD Regular Cab 2012	0.66	0.7	0.68	44
Chevrolet Silverado 1500 Regular Cab 2012	0.56	0.66	0.6	38
Chevrolet Sonic Sedan 2012	0.88	0.64	0.74	44
Chevrolet TrailBlazer SS 2009	0.74	0.7	0.72	37
Chevrolet Tahoe Hybrid SUV 2012	0.83	0.85	0.84	40
Chevrolet Traverse SUV 2012	0.88	0.68	0.77	44
Chrysler PT Cruiser Convertible 2008	0.91	0.6	0.72	48
Chrysler Aspen SUV 2009	0.87	0.91	0.89	43
Chrysler Crossfire Convertible 2008	0.88	0.88	0.88	43
Chrysler 300 SRT-8 2010	0.82	0.93	0.87	45
Daewoo Nubira Wagon 2002	0.88	0.88	0.88	40
Chrysler Sebring Convertible 2010	0.86	0.84	0.85	37
Chrysler Town and Country Minivan 2012	1	0.8	0.89	45
Dodge Caliber Wagon 2007	0.67	0.88	0.76	42
Dodge Caravan Minivan 1997	0.76	0.62	0.68	40
Dodge Caliber Wagon 2012	0.76	0.95	0.85	43
Dodge Challenger SRT8 2011	0.94	0.87	0.91	39
Dodge Charger SRT-8 2009	0.74	0.69	0.72	42

Dodge Charger Sedan 2012	0.77	0.9	0.83	41
Dodge Durango SUV 2007	0.71	0.89	0.79	38
Dodge Durango SUV 2012	0.94	0.8	0.87	41
Dodge Dakota Crew Cab 2010	0.81	0.87	0.84	45
Dodge Dakota Club Cab 2007	0.95	0.86	0.9	43
Dodge Journey SUV 2012	0.95	0.89	0.92	44
Dodge Ram Pickup 3500 Crew Cab 2010	0.85	0.88	0.86	40
Dodge Magnum Wagon 2008	0.94	0.79	0.86	42
Dodge Sprinter Cargo Van 2009	0.7	0.68	0.69	44
Dodge Ram Pickup 3500 Quad Cab 2009	0.74	0.74	0.74	39
Eagle Talon Hatchback 1998	0.9	0.78	0.84	46
Ferrari 458 Italia Convertible 2012	1	0.96	0.98	27
Ford F-150 Regular Cab 2007	0.91	0.94	0.93	33
Ferrari 458 Italia Coupe 2012	0.71	0.74	0.72	39
Ford Edge SUV 2012	0.75	0.64	0.69	42
Ferrari California Convertible 2012	0.94	0.79	0.86	39
FIAT 500 Convertible 2012	0.87	0.79	0.82	42
Ford E-Series Wagon Van 2012	0.89	0.79	0.84	43
FIAT 500 Abarth 2012	0.94	0.92	0.93	37
Ferrari FF Coupe 2012	0.91	0.93	0.92	43
Fisker Karma Sedan 2012	0.83	0.89	0.86	44
Ford Expedition EL SUV 2009	0.84	0.8	0.82	45
Ford Focus Sedan 2007	0.95	0.93	0.94	42

Ford F-450 Super Duty Crew Cab 2012	0.86	0.9	0.88	41
Ford Ranger SuperCab 2011	0.78	0.83	0.8	42
Ford F-150 Regular Cab 2012	0.84	0.82	0.83	45
Ford Freestar Minivan 2007	0.84	0.98	0.91	44
Geo Metro Convertible 1993	0.86	0.71	0.78	45
GMC Acadia SUV 2012	0.77	0.61	0.68	44
Ford Fiesta Sedan 2012	0.81	0.81	0.81	42
Ford Mustang Convertible 2007	0.88	0.82	0.85	44
Ford GT Coupe 2006	0.76	0.65	0.7	40
GMC Terrain SUV 2012	0.7	0.76	0.73	68
GMC Savana Van 2012	0.88	0.93	0.9	41
GMC Canyon Extended Cab 2012	0.85	0.69	0.76	42
Honda Odyssey Minivan 2007	0.87	0.89	0.88	44
GMC Yukon Hybrid SUV 2012	0.94	0.74	0.83	43
Honda Odyssey Minivan 2012	0.84	0.92	0.88	39
Honda Accord Sedan 2012	0.72	0.74	0.73	39
HUMMER H2 SUT Crew Cab 2009	0.63	0.71	0.67	38
HUMMER H3T Crew Cab 2010	0.92	0.85	0.89	41
Honda Accord Coupe 2012	0.76	0.93	0.84	42
Hyundai Elantra Touring Hatchback 2012	0.44	0.83	0.58	24
Hyundai Santa Fe SUV 2012	0.91	0.71	0.8	42
Hyundai Tucson SUV 2012	0.77	0.81	0.79	42
Hyundai Sonata Hybrid Sedan 2012	0.79	0.81	0.8	42
Hyundai Veloster Hatchback 2012	0.74	0.98	0.84	43

Hyundai Azera Sedan 2012	0.97	0.79	0.87	42
Hyundai Genesis Sedan 2012	0.88	0.88	0.88	33
Hyundai Elantra Sedan 2007	0.82	0.95	0.88	39
Hyundai Sonata Sedan 2012	0.72	0.91	0.8	43
Hyundai Accent Sedan 2012	0.77	0.8	0.79	41
Infiniti QX56 SUV 2011	0.9	0.43	0.58	42
Jeep Wrangler SUV 2012	0.85	0.85	0.85	34
Jeep Patriot SUV 2012	0.94	0.91	0.92	32
Infiniti G Coupe IPL 2012	0.58	0.95	0.72	40
Jeep Compass SUV 2012	0.83	0.43	0.57	46
Jeep Grand Cherokee SUV 2012	0.81	0.83	0.82	42
Jaguar XK XKR 2012	0.83	0.87	0.85	45
Hyundai Veracruz SUV 2012	0.88	0.82	0.85	44
Jeep Liberty SUV 2012	0.82	0.82	0.82	44
Isuzu Ascender SUV 2008	1	0.93	0.96	43
Lincoln Town Car Sedan 2011	0.8	0.86	0.83	43
Land Rover Range Rover SUV 2012	0.98	0.91	0.94	44
Lamborghini Reventon Coupe 2008	1	0.8	0.89	35
Land Rover LR2 SUV 2012	0.87	0.94	0.91	36
Lamborghini Gallardo LP 570-4 Superleggera 2012	0.88	0.71	0.79	42
Mazda Tribute SUV 2011	0.85	0.98	0.91	42
Lamborghini Diablo Coupe 2001	0.94	0.87	0.91	39
Lamborghini Aventador Coupe 2012	0.89	0.94	0.92	36
Maybach Landaulet Convertible 2012	0.83	0.86	0.85	29

McLaren MP4-12C Coupe 2012	0.78	0.89	0.83	36
Mercedes-Benz E-Class Sedan 2012	0.77	0.91	0.83	44
MINI Cooper Roadster Convertible 2012	0.94	0.94	0.94	48
Nissan Juke Hatchback 2012	0.79	0.93	0.86	45
Mercedes-Benz S-Class Sedan 2012	0.64	0.37	0.47	43
Nissan 240SX Coupe 1998	0.68	0.91	0.78	44
Mitsubishi Lancer Sedan 2012	0.79	0.72	0.75	36
Mercedes-Benz Sprinter Van 2012	0.75	0.8	0.78	41
Mercedes-Benz C-Class Sedan 2012	0.94	0.64	0.76	47
Mercedes-Benz 300-Class Convertible 1993	0.87	0.89	0.88	46
Mercedes-Benz SL-Class Coupe 2009	0.94	0.66	0.77	44
Rolls-Royce Ghost Sedan 2012	0.95	0.93	0.94	42
Nissan Leaf Hatchback 2012	1	0.82	0.9	38
Nissan NV Passenger Van 2012	0.91	0.89	0.9	44
Rolls-Royce Phantom Sedan 2012	0.69	0.63	0.66	43
Rolls-Royce Phantom Drophead Coupe Convertible 2012	0.78	0.68	0.73	41
Porsche Panamera Sedan 2012	0.76	0.66	0.7	38
Scion xD Hatchback 2012	0.69	0.73	0.71	30
Ram C-V Cargo Van Minivan 2012	0.59	0.66	0.62	44
Plymouth Neon Coupe 1999	0.69	0.8	0.74	41
smart fortwo Convertible 2012	0.75	0.8	0.77	45
Toyota Camry Sedan 2012	0.61	0.64	0.63	42
Toyota Corolla Sedan 2012	0.9	0.68	0.78	38

Tesla Model S Sedan 2012	0.69	0.8	0.74	46
Spyker C8 Convertible 2009	0.78	0.76	0.77	42
Spyker C8 Coupe 2009	0.62	0.57	0.6	40
Suzuki SX4 Sedan 2012	0.69	0.89	0.78	38
Suzuki Kizashi Sedan 2012	0.86	0.93	0.89	40
Toyota 4Runner SUV 2012	0.7	0.88	0.78	43
Suzuki SX4 Hatchback 2012	0.8	0.74	0.77	43
Suzuki Aerio Sedan 2007	0.81	0.79	0.8	38
Volvo XC90 SUV 2007	0.85	0.93	0.89	42
Volvo C30 Hatchback 2012	0.92	0.98	0.95	46
Volkswagen Beetle Hatchback 2012	0.85	0.77	0.8	43
Volvo 240 Sedan 1993	0.91	0.89	0.9	45
Volkswagen Golf Hatchback 2012	1	0.8	0.89	41
Volkswagen Golf Hatchback 1991	0.91	0.74	0.82	43
Toyota Sequoia SUV 2012	0.93	0.97	0.95	40

Fig26. Classification Report

SECTION 6: VISUALISATIONS

During exploratory data analysis several insights were made on the data provided. One of the useful insights was to check if all instances of images share the same location and orientation within the dataset. As it is observed from the graph below, the location of the car within the image varies and there are images where the appearance of the car is too small compared to the image itself.
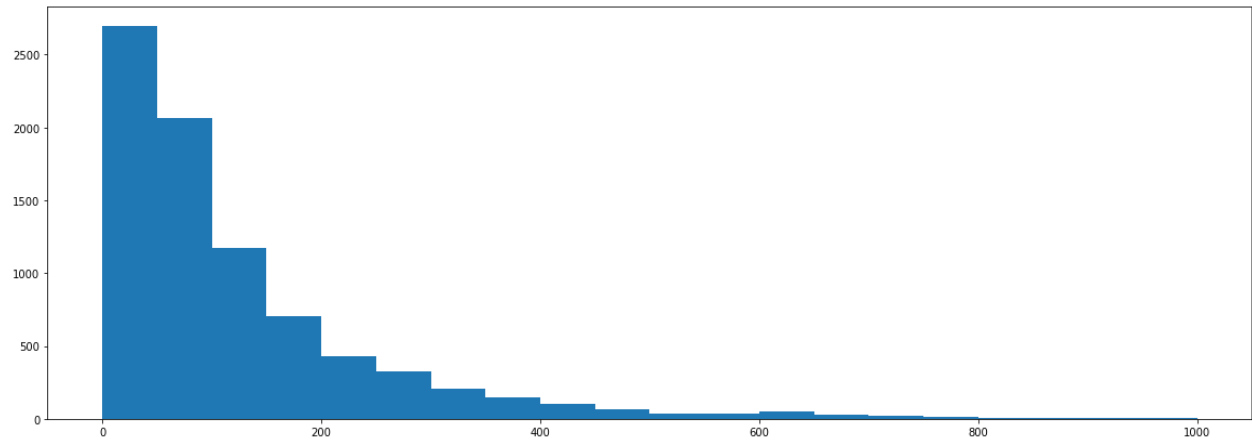
In the above graph, the y-axis shows the number of images (count) and the x-axis shows the orientation of the car in the image.

The Model performance is good if the provided image is having the whole car in the image, some examples are as mentioned below. But if the image is having only the back side of the car or if the image is not having proper brightness or if the image is blurred then it is observed that the performance of the model is poor.

For below mentioned sample images the model performance is good.

For below mentioned sample images the model performance is poor.

SECTION 7: IMPLICATIONS

According to the 2018 Used Car Market Report & Outlook published by Cox Automotive, 40 million used vehicles were sold in the US last year. This represents about 70% of the total vehicles sold. A good portion of these sales already use online resources along various stages of purchasing: searching, pre-qualifying, applying and finally buying. Popular websites for car buyers include AutoTrader.com, Kelly Blue Book, Cars.com, Carvana.com.

Cox Automotive report indicates that most market leaders and Silicon Valley startups speculate that car sales will shift completely to online retailing. This might be an extreme speculation, but these market leaders are interested in providing better user experience while buying a car online, and better recommender systems when a user searches for cars. Peer-to-peer sales platforms like Craigslist, Shift, eBay Motors, etc. are also interested in better fraud detection and monitoring of user postings.

This car image classification system can address these business cases:

1. Ground-truthing of posted used car images on peer-to-peer sales platforms — are these images the cars they specify? Do multiple exterior pics represent the same car?

2. Organizing web-page displays based on the user uploaded images

3. Recommending alternative cars available in the inventory that have similar looks and price

4. In addition, this classification system of cars can help in identifying the fine-grained features important for 3D object detection for self-driving cars.

SECTION 8: LIMITATIONS

Limitations

Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) learn using training data and backpropagation algorithms. A critical step is to fit the AI approach to the problem and the availability of data. Since these systems are "trained" rather than programmed, the various processes often require huge amounts of labeled data to perform complex tasks accurately.

Obtaining large data sets can be difficult.

In some domains, they may simply not be available, but even when available, the labeling efforts can require enormous human resources.

Further, it can be difficult to discern how a mathematical model trained by deep learning arrives at a particular prediction, recommendation, or decision.

A black box, even one that does what it's supposed to, may have limited utility, especially where the predictions or decisions impact society and hold ramifications that can affect individual well-being.

In such cases, users sometimes need to know the reason behind the working, such as why an algorithm reached its recommendations—from making factual findings with legal repercussions to arriving at business decisions, such as lending, that have regulatory repercussions—and why certain factors (and not others) were so critical in a given instance.

Scope for Enhancement

The original dataset has roughly 50–50% train-test split. For further analysis, these images can be combined, and a train-test split of 80–20% can be used.

The dataset after consolidating to five classes has a class imbalance with Vans representing about 6% of the data and on the other extreme Convertible/Coupes and Sedans each representing about 32% of the data. To address this class imbalance the above classification methods can be re-iterated with oversampling techniques: random oversampling, and Synthetic Minority Oversampling TEchnique (SMOTE).

Image segmentation is one of the most popular image processing tasks. Some common pitfalls related to the most frequently used metrics in image segmentation, namely the Dice Similarity Coefficient (DSC), the Harsdorf Distance (HD),and the Intersection over Union (IoU); the problems related to segmentation metrics are assigned to four categories, namely

(1) awareness of fundamental mathematical properties of metrics, necessary to determine the applicability of a metric,
a) Small structures - Segmentation of small structures, such as brain lesions, cells imaged at low magnification or distant cars, is essential for many image processing applications
b) Noise/errors in the reference annotations - Similar problems may arise in the presence of annotation

artifacts. Even a single erroneous pixel
in the reference annotation may lead to a substantial decrease in the measured performance, especially in the case of the HD.
c) Shape unawareness - Metrics measuring the overlap between objects are not designed to uncover differences in shapes. This is an important problem for many applications
d) Oversegmentation vs. undersegmentation - In some applications such as autonomous driving or radiotherapy, it may be highly relevant whether an algorithm tends to over- or under-segment the target structure.

(2) suitability for the underlying image processing task,
a) A common problem is that segmentation metrics, such as the DSC, are applied to detection and localization tasks. In general, the DSC is strongly biased against single objects, therefore not appropriate for a detection task of multiple structures

(3) metric aggregation to combine metric values of single images into one accumulated score
a) In the case of metrics with fixed boundaries, like the DSC or the IoU, missing values can easily be set to the worst possible value. For distance-based measures without lower/upper bounds, the strategy of how to deal with missing values is not trivial. In the case of the HD, one may choose the maximum distance of the image and add 1 or normalize the metric values and use the worst possible value. Crucially, however, every choice will produce a different aggregated value, thus potentially affecting the ranking.

(4) metric combination to reflect different aspects in algorithm validation.
a) A single metric typically does not reflect all important aspects that are essential for algorithm validation. Hence, multiple metrics with different properties are often combined. However, the selection of metrics should be well considered as some metrics are mathematically related to each other. A prominent example is the IoU – the most popular segmentation metric in computer vision – which highly correlates with the DSC – the most popular segmentation metric in image analysis. In fact, the IoU and the DSC are mathematically related. Mutually dependent metrics (DSC and IoU) will lead to the same ranking, whereas metrics measuring different properties (HD) will lead to a different ranking.

SECTION 9: REFLECTIONS

To improve the final accuracy and better f1 score, there was more scope in lines of feature extraction that could be experimented further.

Feature extraction

To detect a car in the image, we need to identify feature(s) which uniquely represent a car. which are robust enough when it comes to changing perspectives and shapes of the object. Here are techniques that we can experiment with to build a more robust feature extraction.

Histogram of Oriented Gradients (HOG)

In order to have a robust feature set and increase our accuracy rate we can use Histogram of Oriented Gradients (HOG). This feature descriptor is much more resilient to the dynamics of the traffic. Scikit-image python library provides us with the necessary API for calculating HOG features.

With a technique called Sliding Windows we can run prediction models on sub-regions of the images divided into a grid. To increase the robustness of this approach, we can add multiple grids which will be traversed by the prediction model, since cars can appear on the image in various sizes depending on its location on the picture space.

Eliminating false positives

In order to improve the accuracy of the final output we can try to find multiple hits for the object of interest in the similar area. This approach is equivalent to creating a heat map.

It can create redundancy with multi-size sliding windows and count the number of times our classifier predicted the vehicle for a given section of the image across all the windows it appears in. Then label objects with overlapping windows using scipy.ndimage.measurements' label function. And then extract the positions of each label by determining the smallest bounding box our detected object could fit in.

Color Spaces

We can also explore the most suitable color space for our configuration, as HOG features across the 3 RGB channels cab be too similar, therefore not generating features with enough variations. We generate the following outputs across a multitude of color spaces: HUV, HLS, YUV, YCrCb, and LAB

Frame Aggregation

To strengthen our pipeline, experiment with smoothening of all detected windows every n frames. To do so, we accumulate all detected windows between frames $(n-1)*f+1$ to $n*f$, where n is a positive scalar that represents the group of frames we are in. Every time we detect a new object on the current or next frames in the group, we check whether we have detected a similar object in the past, and if so, we append the similar object, thus increasing this object's count across multiple frames. At frame $n*f$ we only retain detected objects (and their associated bounding boxes) that have over m detected counts, to improve the double filtering in the pipeline.

Image augmentation

Image augmentation is a method of modifying existing images in order to generate additional data for the model training process, it is a technique for increasing the size of a training dataset artificially by producing changed versions of the images in the dataset.

Deep learning neural network models that are trained on more data become more efficient and augmentation approaches can provide variations of the images that increase the models' capacity to generalize. The Image Data Generator class in the Keras deep learning neural network toolkit allows you to fit models with picture data augmentation.

THANK YOU NOTE

Thanks to Great Learning team for the help to learn AIML and to do this Capstone Project

Many thanks to our Mentor Mr. Jyanth Mahara. His experience in the field of AIML has guided us in learning throughout this course. His practical knowledge gives us a lot of insights to tackle the issues.

CODE

https://github.com/radhamadhavi1608/CNN_capstone_project-main

REFERENCES

https://keras.io/api/applications/

https://arxiv.org/abs/1704.04861

https://arxiv.org/abs/1801.04381

https://arxiv.org/abs/1512.03385

https://keras.io/api/applications/efficientnet/#efficientnetb7-function