



Interim Report

OBJECT DETECTION - CAR

Natesh	reachnatesha@gmail.com
Lakshman Kumar S	lakshmanrsk@gmail.com
Aditi Ainapure	ainapureaditi@gmail.com
ILAMPARITHIMUTHUSAMY	ilamparithims@gmail.com
Kilambi Rama Krishna	krk1961z@gmail.com
Kunal Gupta	guptakunal.cse@gmail.com

Index

Index.....	2
Summary of the Problem Statement, Data, and Findings	3
Problem Statement:.....	3
Abstract:	3
Data	3
• Data provided in format – Car Images and Annotations	3
• Total Records: Train data – 8144; Test data – 8041	3
• Total Image Classes : 196	
• Useful Information	3
Findings.....	3
Summary of the Approach to EDA and Pre-processing.....	5
Model Building	6
Model performance improvement:	8

Summary of the Problem Statement, Data and Findings

Problem Statement:

Design a DL based car identification model.

Abstract:

Computer vision can be used to automate supervision and generate action appropriate action trigger if the event is predicted from the image of interest. For example a car moving on the road can be easily identified by a camera as make of the car, type, colour, number plates etc. We were successful in designing a CNN algorithm that could identify the class of car based on the image and bounding box.

Data:

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.

- 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.

Dataset has been attached along with this project. Please use the same for this capstone project.

Findings:

1. Train and Test Images are present in separate folders.
 - /content/Car Images/Train Images/Jeep Liberty SUV 2012/00545.jpg',
 - '/content/Car Images/Train Images/Jeep Liberty SUV 2012/06639.jpg',
 - '/content/Car Images/Train Images/Jeep Liberty SUV 2012/05637.jpg',
 - '/content/Car Images/Train Images/Jeep Liberty SUV 2012/06331.jpg',
 - '/content/Car Images/Train Images/Jeep Liberty SUV 2012/00394.jpg',

Image classes can be inferred using the folder of the file.

2. Image coordinates are present in annotations files, so we need to map the Image with its bounding box.

Image Name	Bounding Box coordinates				Image class
00001.jpg	30	52	246	147	181
00002.jpg	100	19	576	203	103
00003.jpg	51	105	968	659	145



We have displayed image samples with the bounding boxes and using the co-ordinates with useful metadata information. Clearly, name of the car is matching with that of the bounding box labels from above

EDA and Pre-processing:

1. For simplicity of coding, we have converted the data into a dataframe using the snippet below

```
df.head()
```

	Car Class	Count	Full Path
0	Hyundai Azera Sedan 2012	42	/content/Car Images/Train Images/Hyundai Azera Sedan 2012
0	Ford Expedition EL SUV 2009	45	/content/Car Images/Train Images/Ford Expedition EL SUV 2009
0	Chevrolet Impala Sedan 2007	43	/content/Car Images/Train Images/Chevrolet Impala Sedan 2007
0	Chevrolet Silverado 1500 Classic Extended Cab 2007	43	/content/Car Images/Train Images/Chevrolet Silverado 1500 Classic Extended Cab 2007
0	Hyundai Sonata Hybrid Sedan 2012	34	/content/Car Images/Train Images/Hyundai Sonata Hybrid Sedan 2012

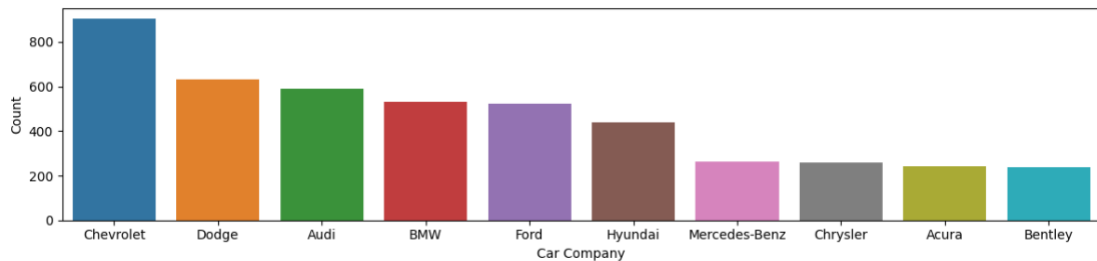
2. Also, try to group the classes in to car companies for understanding if data has some imbalance.

```
df1.describe()
```

	Count
count	49.000000
mean	166.204082
std	185.873947
min	29.000000
25%	44.000000
50%	88.000000
75%	171.000000
max	905.000000

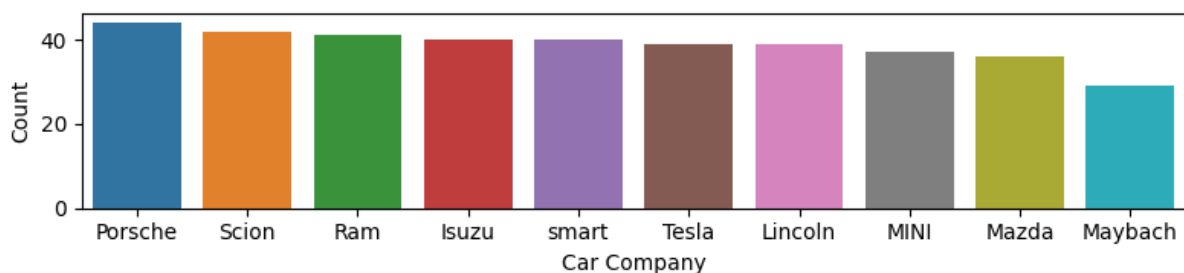
We can see that there are about 49 different companies which makes car. On an average, each company has 166 cars with 905 being maximum and a company making just about 29 cars. More than 75% of these companies have 171 cars. There is some imbalance classes data due the large number of cars available by one single company with 905 cars and a company with just 29 cars. But, for this project we shall not balance this data.

3. Now, having grouped cars into companies, we have plotted top 10 car companies from the given data as below:



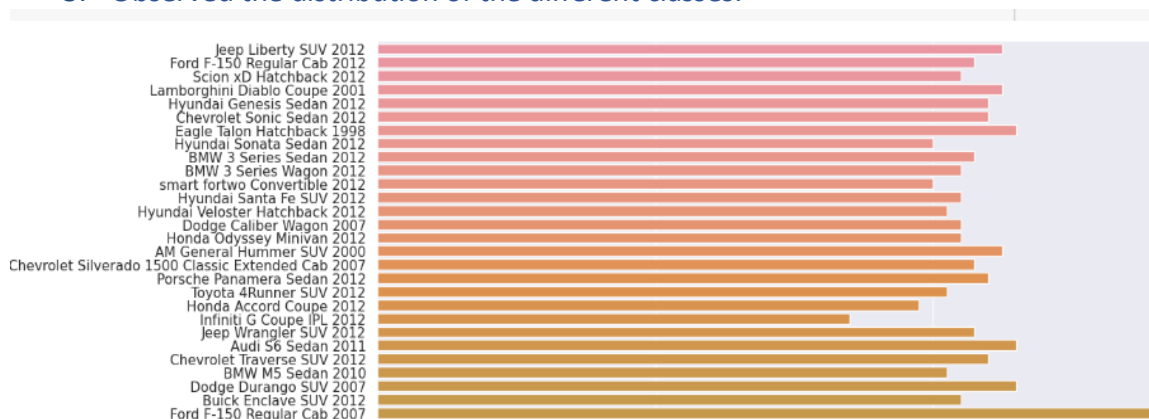
Clearly, Chevrolet is ranked 1 with more than 905 cars and Bentley at #10 with 238 Cars. These 10 cars have average of 218 cars, which is significantly higher than average of 171 cars from all data. This indicates some imbalance and hence let us see top min companies.

4. Now, having grouped cars into companies, we have plotted bottom 10 car companies from the given data as below:



We can clearly see the last 10 companies with Maybach being the least of all with 29 cars and Porsche at 10th position with 44 cars. There is no imbalance in these 10 companies' data, however, from a overall data perspective, there can be some imbalance due to very minimal number compared to top 10 companies.

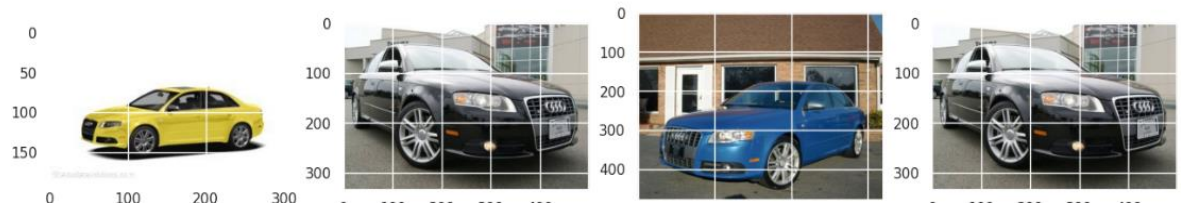
5. Observed the distribution of the different classes.



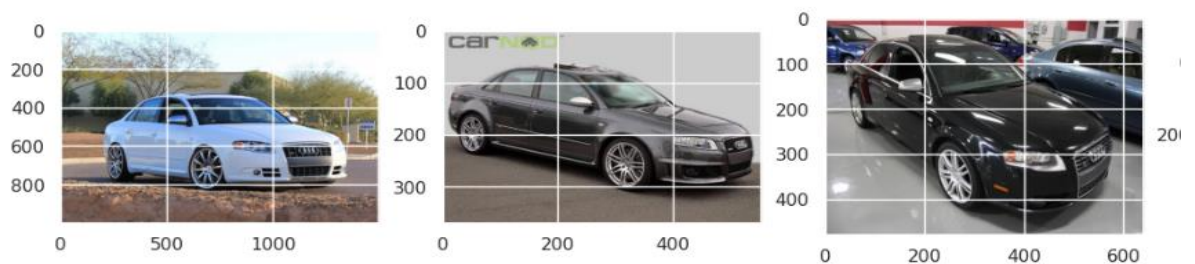
The distribution seems okay with not too many outlier peaks or having very less samples any class.

6. Clearly, we have seen the top 10 companies and last 10 companies
There is a bit of imbalance due to significant contributions from top 10 companies
We can balance the data by removing some classes from top companies, which have minimal cars. we can also ignore last 3 companies with very minimal cars so we can try to balance the data.
7. Note that, for this stage, we are not considering balancing of data but go with data
8. We can see that there are multiple classes with just year being different and hence before we proceed, We shall see if the images in different years are significantly similar

Audi-2012



Audi-2007



9. Clearly, we can see that there is not much changes wrt car design from 2007 model to 2012 cars. We may combine these classes for better predictability.
 - a. Get classes which have only year is different and for that let us remove the year part of the
 - b. Folders and see if folders are same.

1 non_unique_data

	Car Class	Count	Full Path	Car Class without year	
0	Audi S4 Sedan 2007	45	/content/Car Images/Train Images/Audi S4 Sedan 2007	Audi S4 Sedan	
0	Audi S4 Sedan 2012	85	/content/Car Images/Train Images/Audi S4 Sedan 2012	Audi S4 Sedan	
0	Bentley Continental GT Coupe 2007	46	/content/Car Images/Train Images/Bentley Continental GT Coupe 2007	Bentley Continental GT Coupe	
0	Bentley Continental GT Coupe 2012	81	/content/Car Images/Train Images/Bentley Continental GT Coupe 2012	Bentley Continental GT Coupe	
0	Dodge Caliber Wagon 2007	42	/content/Car Images/Train Images/Dodge Caliber Wagon 2007	Dodge Caliber Wagon	

c.

Summary of the Approach to EDA and Pre-processing:

- Checking for Null values and duplicates in training data.
- Verifying that train and test folders have similar no of dataset.
- Assign each of the image to its ClassName(String).
- Using annotation to read the bounding box and numeric class name for each image.
- Verify that the number of classes is same as mentioned in dataset.
- Verify that the bounding box is assigned accurately to its image, bounding box is properly covering the image.
- Group the cars into companies by taking the first-string value of the car class and see which car company has the maximum/min cars in the given dataset
- Plot the top 10 Car companies and bottom 10 car companies from the given data to see imbalance
- Displaying the distribution of different Car classes using Bar Plot to check that distribution among classes is same.
- As car making year is not affecting the car model, so combine the multiple years of model to same class.
- Get classes which have only year is different and for that let us remove the year part of the. Merge the cars with same name just the year part different and verify the count again.
- Verify the images again with their bound box and model name




```
1 # Example usage Train Images
2 random_images = random.sample(list(test_dict.keys()), 5)
3 display_images(random_images, train_dict, "/content/Car Images/Train Images")
```



Model Building:

The data is already split into train and test.

- Use CNN : as this is a problem that can only be solved using neural network.
- Use Softmax for output layer: As we have categorical data, where all the outputs should be one of these 196 categories.
- We are using Adam as optimizer as it is very fast.
- Use Relu for internal layers, as for images we will be having data in only one direction(positive side only)
- As output layer is softmax we are using “categorical cross entropy” for multi-class classification where each example belongs to a single class.
- Model Summary of the base model is shown below

 <code>model.summary()</code>		
Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 222, 222, 16)	448
max_pooling2d (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_1 (Conv2D)	(None, 109, 109, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
flatten (Flatten)	(None, 43264)	0
dense (Dense)	(None, 64)	2768960
dense_1 (Dense)	(None, 196)	12740
=====		
Total params: 2,805,284		
Trainable params: 2,805,284		
Non-trainable params: 0		
=====		

- Model training for 10 epochs is shown below:

```
[80] history = model.fit( train_generator,
                          epochs = 10, batch_size = 32,
                          callbacks=[save_best], verbose=1,
                          validation_data = test_generator)
```

```
Epoch 1/10
255/255 [=====] - 142s 556ms/step - loss: 1.4077 - accuracy: 0.6090 - val_loss: 14.7684 - val_accuracy: 0.0190
Epoch 2/10
255/255 [=====] - 141s 555ms/step - loss: 1.2244 - accuracy: 0.6621 - val_loss: 16.0769 - val_accuracy: 0.0183
Epoch 3/10
255/255 [=====] - 141s 552ms/step - loss: 1.0997 - accuracy: 0.6992 - val_loss: 17.0150 - val_accuracy: 0.0184
Epoch 4/10
255/255 [=====] - 143s 561ms/step - loss: 1.0121 - accuracy: 0.7263 - val_loss: 19.0808 - val_accuracy: 0.0178
Epoch 5/10
255/255 [=====] - 141s 554ms/step - loss: 0.9442 - accuracy: 0.7412 - val_loss: 19.2367 - val_accuracy: 0.0178
Epoch 6/10
255/255 [=====] - 141s 553ms/step - loss: 0.9112 - accuracy: 0.7597 - val_loss: 21.7434 - val_accuracy: 0.0163
Epoch 7/10
255/255 [=====] - 142s 559ms/step - loss: 0.8358 - accuracy: 0.7747 - val_loss: 21.2422 - val_accuracy: 0.0193
Epoch 8/10
255/255 [=====] - 142s 556ms/step - loss: 0.8005 - accuracy: 0.7832 - val_loss: 23.6032 - val_accuracy: 0.0189
Epoch 9/10
255/255 [=====] - 142s 556ms/step - loss: 0.7673 - accuracy: 0.7941 - val_loss: 24.7150 - val_accuracy: 0.0187
Epoch 10/10
255/255 [=====] - 140s 551ms/step - loss: 0.7381 - accuracy: 0.8032 - val_loss: 23.2190 - val_accuracy: 0.0221
```

- Model learning rate and Test accuracy is as below:

```

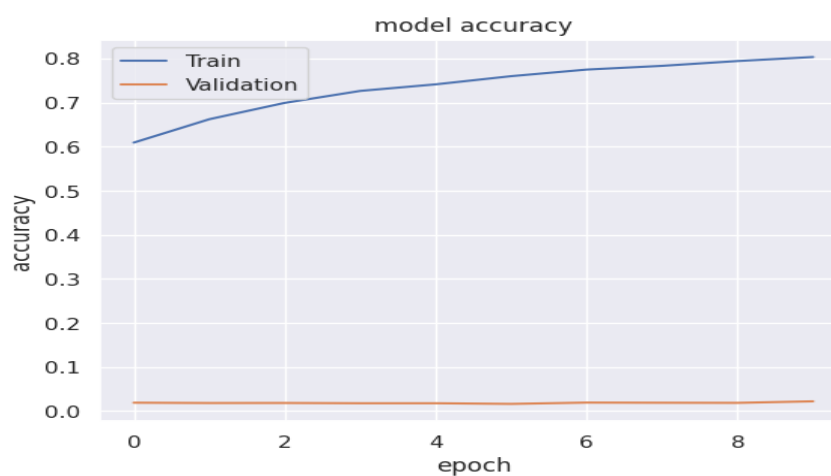
✓ 1m test_loss, test_acc = model.evaluate(test_generator)
      print('Test loss:', test_loss)
      print('Test accuracy:', test_acc)

252/252 [=====] - 70s 278ms/step - loss: 23.2190 - accuracy: 0.0221
Test loss: 23.219026565551758
Test accuracy: 0.022136550396680832

```

[+ Code](#) [+ Text](#)

- Plotted Model accuracy and model loss



Conclusion :

- With 10 epochs and basic few layers, our basic CNN model has a test accuracy of 2.2% and definitely this approach is not at yielding anything for us. We need to effectively use the annotations to mask and use a different model for better accuracy

Next Steps: Model performance improvement:

1. Data Augmentation

Having a large dataset is crucial for the performance of the deep learning model. However, we can improve the performance of the model by augmenting the data we already have. It also helps the model to generalize on different types of images. In data augmentation, we add different filters or slightly change the images we already have for example add a random zoom in, zoom out, rotate the image by a random angle

2. Changing the number of layers and playing with number epochs may give us better accuracy.
3. Transfer Learning: Pretrained models like VGG, ResNet, and Inception are trained on large datasets like ImageNet and can be used as a starting point for training the model on the car images dataset. By using transfer learning, we can leverage the knowledge learned by these models on similar image recognition tasks.
4. Regularization: Techniques such as dropout and L1/L2 regularization can be used to reduce overfitting and improve the generalization ability of the model.
5. Hyperparameter Tuning: Hyperparameters such as learning rate, batch size, and optimizer can have a significant impact on the performance of the model. Grid search or random search can be used to find the optimal set of hyperparameters for the model.
6. Ensemble Learning: This involves combining the predictions of multiple models to improve the overall performance. A simple way to implement ensemble learning is to train multiple base CNN models with different initialization and then average their predictions.
7. By applying these techniques, we can improve the accuracy and generalization ability of the base CNN model

