AIML CAPSTONE PROJECT

Computer Vison Car Detection

PROJECT MEMBER

SR NO	NAME	EMAIL ID
1	Rupali Botre	botrerupalid@gmail.com
2	Bama Charan Kundu	bamachrn@gmail.com
3	Karthikeyan C	
4	Chanchal Singh Adhikari	

TABLE OF CONTENTS

1. PROJECT DESCRIPTION

DOMAIN: Automotive. Surveillance.

CONTEXT:

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.

DATA DESCRIPTION:

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. Original link to the dataset: https://www.kaggle.com/jutrera/stanford-car-dataset-by-classes-folder

Reference: 3D Object Representations for Fine-Grained Categorisation, Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei 4th IEEE

Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.

PROJECT OBJECTIVE:

Design a DL based car identification model.

2. MILESTONE 1

2.1 Summary of Problem Statement

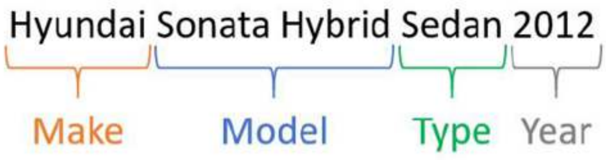
The Stanford Car Dataset will be utilized to build a vehicle recognition predictive model. The goal of the model is to classify a car's year, make and model given an input image.

2.2 Approach to EDA and Pre-processing

- The Kaggle Stanford Cars Dataset contains total 16,185 images of cars. There are a total of 196 classes of cars in this dataset. The data is split in half to be used as training and testing sets. The data also comes with class labels and bounding boxes for all images.
- The classes are typically at the level of Year, Make and Model (e.g. 2012 Tesla Model S or 2012 BMW M3 coupe). The sizes of each image are different. Utilization of the bounding boxes is essential in the pre-processing phase to first obtain images that focus on the objects of interest, which in this case are the vehicles. The actual images are in JPG format, but the data comes zipped in TGZ/TAR format.
- The Stanford Car Dataset will be utilized to build a vehicle recognition predictive model.
- The goal of the model is to classify a car's year, make and model given an input image.
- The dataset contained no missing values, so no imputations or data removal was required due to the nature of image data. In terms of Exploratory Data Analysis, the class labels were split to explore the individual Make, Model, Type and Year levels of the labels.
- The string-formatted labels were split by a space, then the output of that were categorized into the Make, Model, Type and Year levels. This was tricky, since some the Make and Model levels had different lengths (for instance, Aston Martin vs. BMW in the Make level and Sonata vs. F-450 Super Duty Crew in the Model level). This extraction of class label levels was performed to the best of our abilities.
- There were 196 classes originally. Because of this high total class number, the levels of class labels were analysed with the hopes of reducing the total class number. Initially the class labels were analyzed by human eyes.
- While the Stanford dataset contained pre-split training and testing data, 50:50 data.

2.3 Data and Findings

- The original dataset defined a 'class' as the combination of make, model, and year.
- This yields 196 individual and unique classes. An example of one of these classes is shown in Figure. The class levels were parsed into the components also shown in Figure. It may be possible to extract more useful information by separating these characteristics.



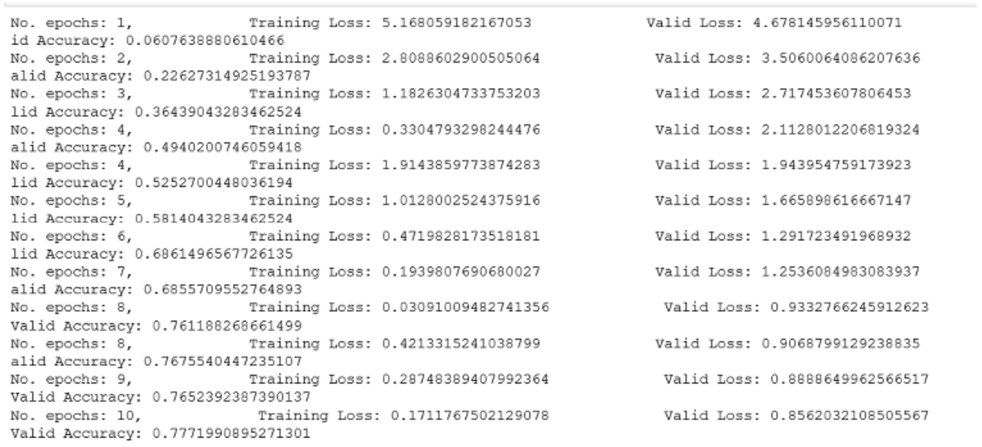
- The following table provides specific descriptive statistics from the entire original dataset.
- The image dimensions (height, width, and channels) were added to support future modelling decisions. Due to the way that the image of this dataset was created, a thorough Exploratory Data Analysis of the original class distributions was highly desired.

	ClassNo	Class	Make	Model	Type	Year	Image Height	Image Width	Color Channels
Type	Integer	String	String	String	String	Integer	Integer	Integer	Integer
Uniques	196	196	49	177	13	16	Several	Several	3
Mean	-	-	-	-	-	2009.56	308	573	-
Std Dev	-	-	-	-	-	4.43	214	375	-

3. MILESTONE 2

3.1 RESNET

- The initial custom convolutional neural networks with 1~3 ConvNet layers (conv-conv-pool) with relu activations fully connected layers performed poorly. These custom CNNs was trained after tuning the learning rate according to the One-Cycle-Policy by Leslie and fit on regular Stanford Cars images (under-sampled). The 3rd model (3 ConvNet layers) had the highest validation accuracy of about 5% post learning rate tuning.
- These initial set of models illustrated that simple CNN models will not be able to perform well on this complex and non-structured car image data. Since these initial models did not perform well.
- Custom convolutional neural networks (CNNs) were focused on predicting the Vehicle Make level. The following plot shows the training and validation accuracies (red and blue) along with the Top-5 accuracies (green and black). The top-5 accuracy scores show that the network has more success at proposing the five most-likely manufacturers instead of picking just one.
  It may be possible to leverage this model as a type of autoencoder for a separate downstream model to make better classifications.
- Some model-specific image pre-processing is done prior to training. The model is trained in batches using augmented images since it is believed that there may not be enough data for the large number of target classes proposed. Image augmentation has greatly improved the rate at which models converge on a solution and the maximum accuracy overall.
- These models are trained on large visual databases, which are often used in object recognition. Since these pre-trained models are trained on large visual databases, the first step of implementing these models are adding custom fully connected layers. Once the top layers (FC layer) are added, the model has to be frozen (setting the convolution layers' trainable status to False) in order to only train the top layers. This process makes these pre-trained models suitable for any image data.
- The initial setup for the ResNet34 model was done utilizing module. Which had multiple BatchNorms, Dropouts and Dense layer with linear and relu activations.

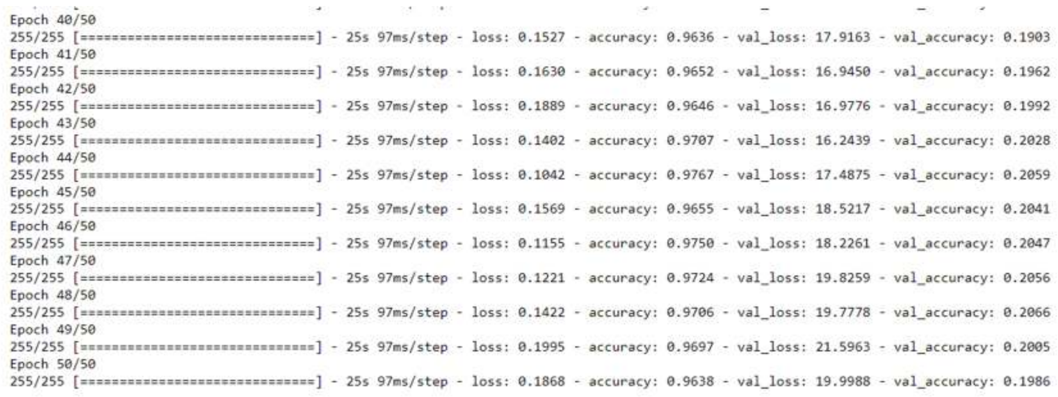


```
No. epochs: 1,             Training Loss: 5.168059182167053            Valid Loss: 4.678145956110071
id Accuracy: 0.0607638880610466
No. epochs: 2,             Training Loss: 2.8088602900505064            Valid Loss: 3.5060064086207636
alid Accuracy: 0.22627314925193787
No. epochs: 3,             Training Loss: 1.1826304733753203            Valid Loss: 2.717453607806453
lid Accuracy: 0.36439043283462524
No. epochs: 4,             Training Loss: 0.3304793298244476            Valid Loss: 2.1128012206819324
alid Accuracy: 0.4940200746059418
No. epochs: 4,             Training Loss: 1.9143859773874283            Valid Loss: 1.943954759173923
lid Accuracy: 0.5252700448036194
No. epochs: 5,             Training Loss: 1.0128002524375916            Valid Loss: 1.665898616667147
lid Accuracy: 0.5814043283462524
No. epochs: 6,             Training Loss: 0.4719828173518181            Valid Loss: 1.291723491968932
lid Accuracy: 0.6861496567726135
No. epochs: 7,             Training Loss: 0.1939807690680027            Valid Loss: 1.2536084983083937
alid Accuracy: 0.6855709552764893
No. epochs: 8,             Training Loss: 0.03091009482741356            Valid Loss: 0.9332766245912623
Valid Accuracy: 0.761188268661499
No. epochs: 8,             Training Loss: 0.4213315241038799            Valid Loss: 0.9068799129238835
alid Accuracy: 0.7675540447235107
No. epochs: 9,             Training Loss: 0.28748389407992364            Valid Loss: 0.8888649962566517
Valid Accuracy: 0.7652392387390137
No. epochs: 10,             Training Loss: 0.1711767502129078            Valid Loss: 0.8562032108505567
Valid Accuracy: 0.7771990895271301
```

3.2 Mobile Net

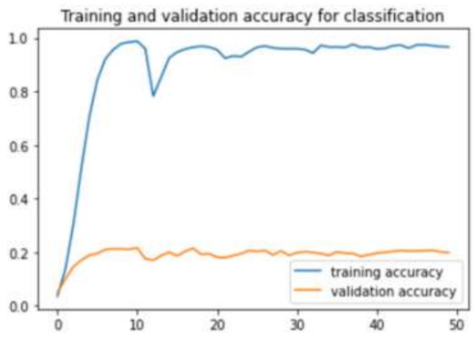
We shall be using Mobilenet as it is lightweight in its architecture. It uses depthwise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it.

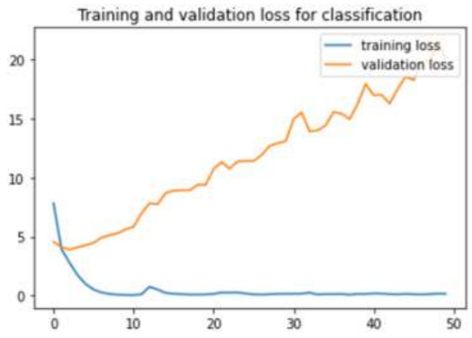
- We have used the MobileNetV2 model with weights - ‘imagenet’ and Activation Function as Softmax.
- Used two Dense Layers 1024 Relu and 196 Sofmax.
- Created Bounding Box model.
- Created IOU function for finding out the difference between provided bounding box and predicted bounding box.
- For Bounding Box Model, we achieved 19% accuracy with 50 Epoch.



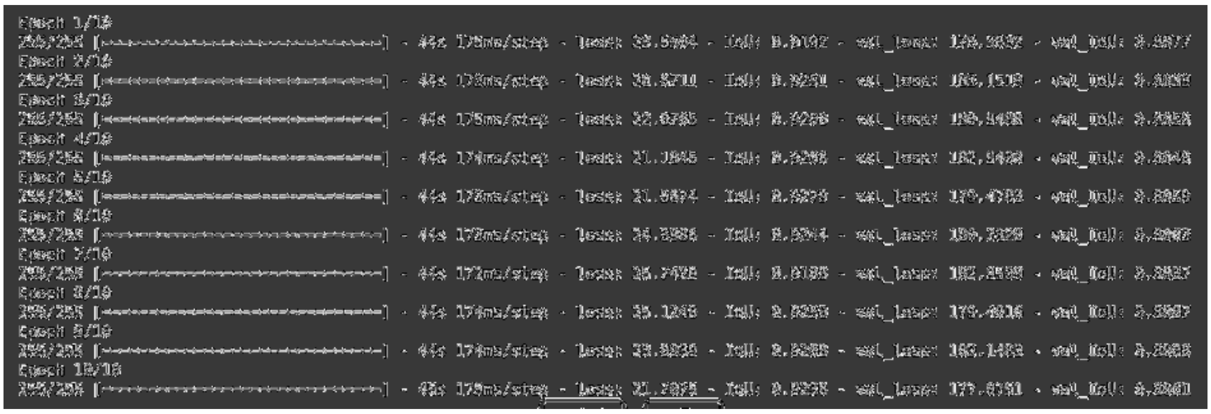
```
Epoch 40/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1527 - accuracy: 0.9636 - val_loss: 17.9163 - val_accuracy: 0.1903
Epoch 41/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1630 - accuracy: 0.9652 - val_loss: 16.9450 - val_accuracy: 0.1962
Epoch 42/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1889 - accuracy: 0.9646 - val_loss: 16.9776 - val_accuracy: 0.1992
Epoch 43/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1402 - accuracy: 0.9707 - val_loss: 16.2439 - val_accuracy: 0.2028
Epoch 44/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1042 - accuracy: 0.9767 - val_loss: 17.4875 - val_accuracy: 0.2059
Epoch 45/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1569 - accuracy: 0.9655 - val_loss: 18.5217 - val_accuracy: 0.2041
Epoch 46/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1155 - accuracy: 0.9750 - val_loss: 18.2261 - val_accuracy: 0.2047
Epoch 47/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1221 - accuracy: 0.9724 - val_loss: 19.8259 - val_accuracy: 0.2056
Epoch 48/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1422 - accuracy: 0.9706 - val_loss: 19.7778 - val_accuracy: 0.2066
Epoch 49/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1995 - accuracy: 0.9697 - val_loss: 21.5963 - val_accuracy: 0.2005
Epoch 50/50
255/255 [==============================] - 25s 97ms/step - loss: 0.1868 - accuracy: 0.9638 - val_loss: 19.9988 - val_accuracy: 0.1986
```

- We have achieved following Training Validation Accuracy and Loss as shown in following graph:





- We have used the MobileNetV2 model with weights from - ‘imagenet’ and Activation Function as Softmax.
- For bounding box we have taken the output of top layer from mobilenet
- Added two Layers on top of it
    - 2Dconv layer with kernel size 7X7 naming coords. This layers gives the output as (1,1,4) array
    - Reshape layer to generate co-ords for the bounding box.
- We have trained classification model with 10 epoch resulting maximum of 80% validation IoU and 92% training IoU.
- Created IOU function for finding out the difference between provided bounding box and predicted bounding box.



```
Epoch 1/10
255/255 [==============================] - 44s 175ms/step - loss: 33.5904 - IoU: 0.9192 - val_loss: 178.3832 - val_IoU: 0.8877
Epoch 2/10
255/255 [==============================] - 44s 173ms/step - loss: 30.8710 - IoU: 0.9251 - val_loss: 166.1519 - val_IoU: 0.8083
Epoch 3/10
255/255 [==============================] - 44s 175ms/step - loss: 22.6785 - IoU: 0.9258 - val_loss: 190.5498 - val_IoU: 0.7858
Epoch 4/10
255/255 [==============================] - 44s 174ms/step - loss: 21.1845 - IoU: 0.9298 - val_loss: 182.9428 - val_IoU: 0.8048
Epoch 5/10
255/255 [==============================] - 44s 172ms/step - loss: 21.5874 - IoU: 0.9279 - val_loss: 179.4783 - val_IoU: 0.8059
Epoch 6/10
255/255 [==============================] - 44s 172ms/step - loss: 24.3988 - IoU: 0.9244 - val_loss: 196.3329 - val_IoU: 0.8067
Epoch 7/10
255/255 [==============================] - 44s 172ms/step - loss: 25.7428 - IoU: 0.9185 - val_loss: 182.2598 - val_IoU: 0.8027
Epoch 8/10
255/255 [==============================] - 44s 174ms/step - loss: 25.1248 - IoU: 0.9255 - val_loss: 179.4816 - val_IoU: 0.5587
Epoch 9/10
255/255 [==============================] - 44s 174ms/step - loss: 23.8235 - IoU: 0.9288 - val_loss: 163.1483 - val_IoU: 0.7868
Epoch 10/10
255/255 [==============================] - 45s 175ms/step - loss: 21.7875 - IoU: 0.9298 - val_loss: 177.6751 - val_IoU: 0.8061
```

    - We have got the following matrix for the IoU and loss