Topic:- Train a light weight model to identify make and model of the cars.

Dataset - http://ai.stanford.edu/~jkrause/cars/car\_dataset.html

Questions –

1. Find out the make and model that is most represented and the one that is least represented class? Is there a class imbalance problem, how would you handle class imbalance if it were to exist (2 marks)

**Response:**

**a) The make model GMC Savana Van 2012 is most represented and Hyundai Accent Sedan 2012 is least represented.**

**b) Yes, as per the distribution of data in different classes, there is a class imbalance. While working on the problem statement, I tried a couple of approaches as below:**

**i)I tried to resample the minority class identified using Pytorch WeightedRandomSampler. ii) I tried to enhance the categorical crossentropy loss function by considering Focal Loss based on FAIR’s**[**Focal Loss for Dense Object Detection**](https://arxiv.org/pdf/1708.02002.pdf)**research paper(**[**https://arxiv.org/pdf/1708.02002.pdf**](https://arxiv.org/pdf/1708.02002.pdf)**).**

**Details and Distribution:**

The corresponding class information is as below:

**Class 135(if zero indexed from class 0) is least represented and class 118(if zero indexed from class 0)) is most represented.**

**Class 136(if no zero indexing) is least represented and class 119(if no zero indexing)) is most represented.**

Chart, bar chart

Description automatically generated

**Distribution of class counts**

Below is a count plot where X axis image count of different classes and y axis gives the count of number of classes (out of 196 classes). The least represented class is having 24 images(count) and the most represented class is having 68 images(count).

Chart, histogram

Description automatically generated

Most classes are having around 45 images.

A picture containing table

Description automatically generated

count 196.00000

mean 41.55102

std 4.33382

min 24.00000

25% 39.75000

50% 42.00000

75% 44.00000

max 68.00000

Name: car\_class, dtype: float64

Int64Index([118, 78, 166, 160, 55, 143, 193, 97, 19, 190,

...

25, 141, 44, 9, 1, 174, 63, 157, 98, 135],

dtype='int64', length=196)

Out[90]:

Text(0.5, 1.0, 'Distibution of class counts')

Reference csv files:

**** ****

1. Apply the image processing techniques and explain the benefits of those technique (2 marks)

**Response:**

There are various image processing techniques like image acquisition, image preprocessing and enhancement, image restoration, morphological processing(erosion,dilation), image segmentation(example RCNN ), object detection(example YoLo), image data compression(lossy ,lossless, examples fractal,wavelets),etc.

Application in context of this problem (I mainly applied preprocessing & enhancement technique):

a) Image preprocessing and enhancement involved **normalization or rescaling** of the pixel values. Some benefits of this step are:

i) images have equal contribution in determining the loss and doesn’t depend on the pixel intensity of the image.

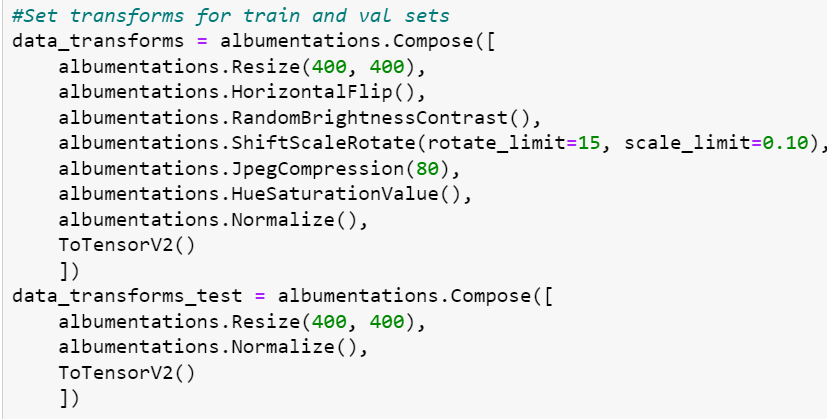
ii) It standardizes the learning rate so that learning rate doesn’t vary based on high and low pixel values of the image.

b)During preprocessing, **data augmentation** was also applied like horizontal flipping, rotation, cropping(random subset of original image which is resized to size of original image), zoom, shifting(shifts the image pixels),brightness(changing image contrast), etc. In context of the problem statement, vertical flipping was **not** done as per domain(car is on a road). The benefit is that it leads to better model performance.


Description automatically generated

c)We can also **standardize** the images for consistency by using featurewise\_center = True and featurewise\_sd\_normalization = True.

d)I also tried the albumentations package for the data transforms.



1. Train a lightweight model, show train and validation loss curves, show steps taken to tune hyper params (1 marks)

a)Training and validation loss curves

**Resnet34+fast ai:**

**The X-axis denote the epoch and Y-axis denotes the loss.**

**Resnext50 +fastai**

Training in progress in Google Colab.(Taking time)

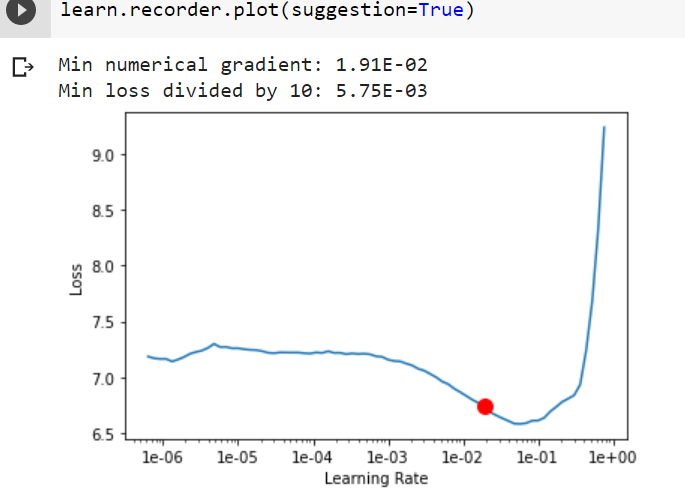
**The X-axis denote the epoch and Y-axis denotes the loss.**

**Keras:**

Training couldn’t complete in google colab(GPU,TPU).

**b) Steps for Hyperparameter tuning:**

i)Adjusting the learning rate



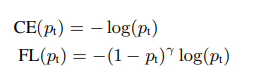
ii)Choosing an optimizer (like sgd, adam), loss function(categoricalcrossentropy for multiclassification with focal loss adjustment in case of class imbalance), activation(multiclass in case of multiclassification)

iii)Deciding batch size and number of epochs.

iv)Techniques like Dropout applied to prevent overfitting.

1. Determine metric to evaluate performance of the model. Report how the model is performing on the metric (2 Marks)
2. **metric for evaluation**

Accuracy was the metric used for evaluation as the class imbalance was addressed by reshaping the standard cross entropy loss using **focal loss** adjustment. This down-weights the loss assigned to well-classified examples and focuses training on a sparse set of misclassified hard examples.





**If loss function is not adjusted with focal loss for class imbalance, metrics like precision, recall and F1 score can be used as evaluation metric.**

Text

Description automatically generated

b) **Performance:**

Accuracy was around 60% for the few epochs which were executed with computational constraints and will increase more with more epochs.

**Resnet34:**

****

**Resnext50:**

Graphical user interface

Description automatically generated with medium confidence



1. Deploy (3 marks)
   1. As a rest api endpoint

i)Save the model with necessary imports

import pickle

file = open("rnext.pkl", "wb") # opening a new file in write mode

pickle.dump(learn2, '/content/drive/MyDrive/rnet50.pkl')

ii)import flask

from flask import Flask

iii)Use flask to deploy as rest api endpoint.

The below code takes input in a POST request through <https://localhost:8081/predict> and returns the prediction in a JSON response.

# Flask constructor takes the name of

# current module (\_\_name\_\_) as argument.

app = Flask(\_\_name\_\_)

# The route() function of the Flask class is a decorator,

# which tells the application which URL should call

# the associated function.

# ‘/predict’ URL is bound with predict() function.

@app.route('/predict', methods=['POST'])

def predict():

     json\_ = request.json

      query\_df = pd.DataFrame(json\_)

      query = pd.get\_dummies(query\_df)

     classifier = joblib.load('rnext.pkl')

      prediction = classifier.predict(query)

      return jsonify({'prediction': list(prediction)})

# main driver function

if \_\_name\_\_ == '\_\_main\_\_':

    # run() method of Flask class runs the application

     # on the local development server

      app.run(port=8081)

* 1. Mobile app model

**Steps while using keras:**

i)Save the model

learn2.save('/content/drive/MyDrive/models/rnext50')

ii)load the model

from keras.models import load\_model

import tensorflow as tf

model = load\_model("/content/drive/MyDrive/models/rnext50.h5")

iii)use TFLiteConverter from tensorflow to create tflite file.

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

tflite\_model = converter.convert()

print("model converted")

iv) # Save the model.

with open('model.tflite', 'wb') as f:

   f.write(tflite\_model)

vi) #Inference to test the tflite model by selecting random image data from the test dataset. Before using the tflite Interpreter, basic operations have to be done to reshape the image to proper format. Code shared below.

****

vii)Install Android studio and build the frontend.