TYRE LIFE PREDICTION SYSTEM

SHORT DESCRIPTION

ABSTRACT

Tyre is the only part of the vehicle which is in contact with the road. The tyre tread depth has a strong influence on the braking performance of the vehicle and should be inspected periodically for driving safety. Worn tyres will result in slippage, a longer braking distance, and even flat tires, which could eventually result in a driving accident. If there is an automated inspection method that will allow the vehicle users to monitor their tyre life with their own hands instead of getting the vehicle to the garage spending time to diagnose it. The prediction involves capturing the image of the thread section of the tyre to be predicted. Then, the image will be processed to extract the thread profile and tread depth from the image. Another image of the sidewall of the same tyre will be captured, which gives details such as tyre brand, model, and size. The depth will be calculated and information will be processed to provide the prediction output in Kilometers. Tyres play a key role in vehicle safety, resolving such type of issues we need such tools and techniques. At the end of the day such systems will boost the confidence while driving and save significant money by extracting maximum out of the tyres. It is absolutely essential that you provide the correct information about the vehicle usage and other parameters. Providing wrong/insufficient information will lead to the wrong estimation and completely wrong recommendations, the model developed can detect various tire defect parameters including sidewall cracking. We intend to include more sophisticated methods to find out the accurate thread depth from tyre images so that the overall prediction can be improved.

<u>LONG DESCRIPTION</u>

INTRODUCTION

The tyres are the most crucial part of every vehicle. The worn-out tyres have many risk factors involved. Tread alone is not the one deciding the life of the tyre but also, The new tyres could be defective sometimes due to the manufacturing process and tyres could also get damaged due to bad roads like sidewall damage that could lead to tyre change. Another thing is tyre wear this could happen two ways Normal wear and Uneven wear, Normal wear is due to normal vehicle wear and tear the Uneven wear is due to improper alignment, worn out suspension, and also due to over-inflation or underinflation. The tyre will deteriorate once it is exposed to the atmosphere tyre due to this age is also one deciding the tyre life other than that the high-speed operation and overloading also causes of irregular tyre wear. Currently, there is no automated system to identify the defects in a vehicle tyre and also to predict the tyre life. At present life prediction of tyres involves penny test for tread depth and other defects by naked eye judgment. With our method, we propose automated testing for defects and also predict tyre life. For automated defect identification, we use Convolutional Neural Network (**CNN).**

1.1 OBJECTIVE OF THE PROJECT:

The tyre life prediction system uses the Convolutional Neural Network **(CNN)** defect identification and life prediction. Our System is trained to identify common tyre defects and we provide recommendations based on the predicted results to improve the tyre life so that the user will be able to ensure safety at the same time they can save the money investing in a new tyre. The system is designed simply to use this can be used from the user's mobile phone itself. They need not bring their vehicle for a garage place for their prediction

LITERATURE SURVEY

2.1 REVIEW BASED ON MEASUREMENT OF TIRE TREAD DEPTH WITH IMAGE TRIANGULATION BY SHIH-YEN HUANG

This system design is used to determine the tyre tread depth using machine vision backed by image triangulation. This method is a non-contact measurement prototype this one gets the distance between the tyre and camera lens such that the depth of the tread will be constructed. There is an Android app which will be used to record the result in user's smartphone. To measure the tread depth users, need to bring their vehicle to the tyre shops with these facilities and tyres needed to be unmounted and placed on the camera base for the tread measurement. This system does not have any additional features and only gives tread depth details.

2.2 THE STUDY ON TIRE TREAD DEPTH MEASUREMENT METHOD BASED ON MACHINE VISION BY XI-BOWANG, AI-JUAN LI, QIN-PENG C

This system uses machine vision to determine the tyre tread depth accurately. In this method, the tyre to be measured is being over a measuring device that captures the tyre tread section with the help of the laser plane. Then, the tread grooves on the profile curve were identified and their positions are determined by an algorithm. Finally, the depth of each groove was calculated successively. This system also measures the tyre tread depth alone but with the help of laser involved to provides results with greater accuracy. The vehicle with tyre to be measured for tread depth should be bought to the garage with this system installed and to be moved over the test platform to perform the measurement.

2.3 DETECTION OF TYRE WEAR OUT USING OPENCV AND CONVOLUTION NEURAL NETWORKS BY ALLIPILLI HARSHITHA AND SAMYUKTHA SAMALA

This is a survey paper in the authors have considered various parameters which determine the lifetime of the tyre. Considering the parameters, the image of the tyre to be predicted is given to the model that uses a convolutional neural network to find the contours in the image.

2.4 A DICTIONARY-BASED METHOD FOR TIRE DEFECT DETECTION YUANYUAN XIANG, CAIMING ZHANG, AND QIANG GUO

New tire defect detection algorithm based on dictionary representation. The dictionary learned from normal images is efficient to represent defect-free images while it has low efficiency to represent defect images due to its capability of capturing key information. Unlike the conventional iterative solution with complicated calculation, the representation coefficients are obtained by multiplying the pseudo-inverse matrix of the learned dictionary and image patch. Automatic visual inspection plays an important role in quality control in the modern tire industry. At present, defect inspection suffers from both low efficiency and high labor intensity and cannot meet the needs of high quality mass manufacturing. Application of computer vision techniques to tire defect detection can avoid the limitations of employing human inspectors

2.5 AN EFFICIENT INDUSTRIAL SYSTEM FOR VEHICLE TYRE (TIRE) DETECTION AND TEXT RECOGNITION USING DEEP LEARNING BY WAJAHAT KAZMI, IAN NABNEY, GEORGE VOGIATZIS, PETER ROSE, AND ALEX CODD

This system is designed to capture the tyre sidewall in low contrast areas during motion. The circularity of the tyres is detected using Hough transform with dynamic radius detection. Then the captured/detected tyres arches are unwrapped into rectangular patches. A cascade of CNN (convolution neural network) is then applied for text recognition. When installed in driveways this system can read tyres moving under a speed of 10mph. The system produced results are accurate, reliable, and efficient and show promise for the intended application.

2.6 TIRE TYPE RECOGNITION THROUGH TREADS PATTERN RECOGNITION AND DOT CODE OCR BY TASNEEM WAHDAN, GHEITH A. ABANDAH, ALIA SEYAM, ALAA AWWAD

In this system tyre text and geometry are scanned and the tyre type is identified and classified according to the code printed in the sidewall which is called the DOT Code. Then the detected tyres are checked for treads and analyze whether the tyre is defective are not. The classification that works on tyre code is more efficient than using DOT Code, but using both is a powerful and accurate classification process.

2.7 TIRE DEFECT DETECTION USING FULLY CONVOLUTIONAL NETWORK BY REN WANG, QIANG GUO, SHANMEI LU1, AND CAIMING ZHANG

In this paper, the authors focus on the application of Deep learning in the industrial field and propose a system based on a full convolution network (FCN) for detecting defects in X-Ray tyre images. The system consists of three phases. The first phase of the system is using a traditional deep network, which extracts the features of tyre images, and feature maps are obtained in the last layer. By replacing fully connected layers with convolution layers, the final feature map consists of sufficient spatial information. After the first two phases, the application develops the coarse segmentation results and refine them through fusing multi-scale feature maps. The outcomes show the proposed method can accurately locate defects in tyre images.

2.8 TIRE DEFECT DETECTION USING IMAGE COMPONENT DECOMPOSITION BY QIANG GUO AND ZHENWEN WEI

This method uses the image decomposition method with total variation filtering this is implemented to overcome the drawbacks of the wavelet-based method. This system consists of three components: texture, background, and defect. Therefore, we use two different filtering methods to successively separate the texture and background components from the defective image.

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Currently, there is no implemented system in India to predict tyre life. The penny test method is commonly used to find the tread depth of the tyre other than that there is no system for defect detection or life prediction automatically.

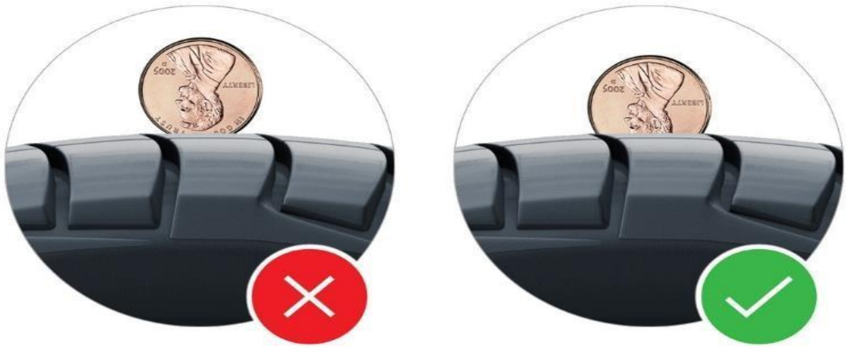


Fig 3.1: Penny Test Method

3.2 PROPOSED SYSTEM

The proposed method consists of two modules, the defect detection module consists of CNN trained image classification model and will be able to classify the given data in one of three pre-trained classification models. We have classified tyres into three major categories: Good, Normal, and Bad. Good Category consists of new tyres, Properly inflated tyres, Tyres without Alignment Problems. Normal categories include tyres without major problems and Bad Category includes tyres with major defects like Tread worn out, sidewall bulge, tread and sidewall crack.

The second module is a result enhancement module that contains an android app. By installing the Tyre Life Prediction System (TLPS) app on their mobile phones the users will be able to predict their vehicle tyre life then and there 24/7. The app gets image input from the user for defect detection and also crucial information like Inflation interval, alignment checking interval, Average speed, and usage environment is collected and processed for precise life prediction of the tyre.

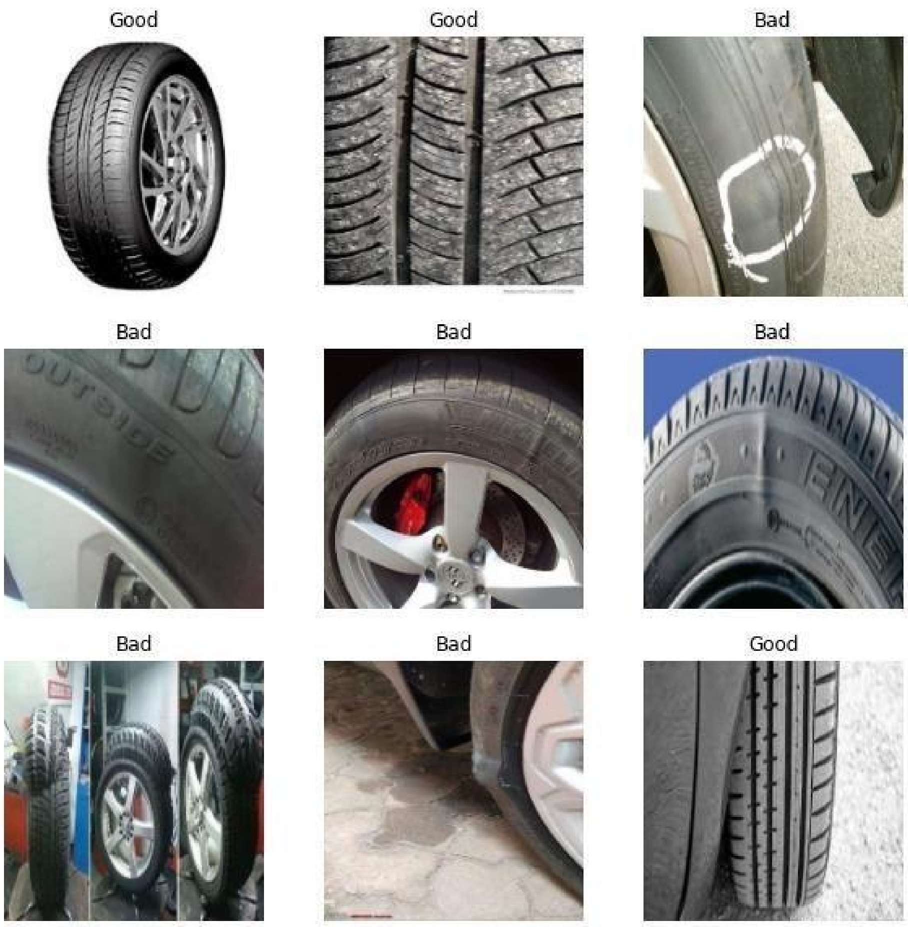


Fig 3.2: Classification Categories

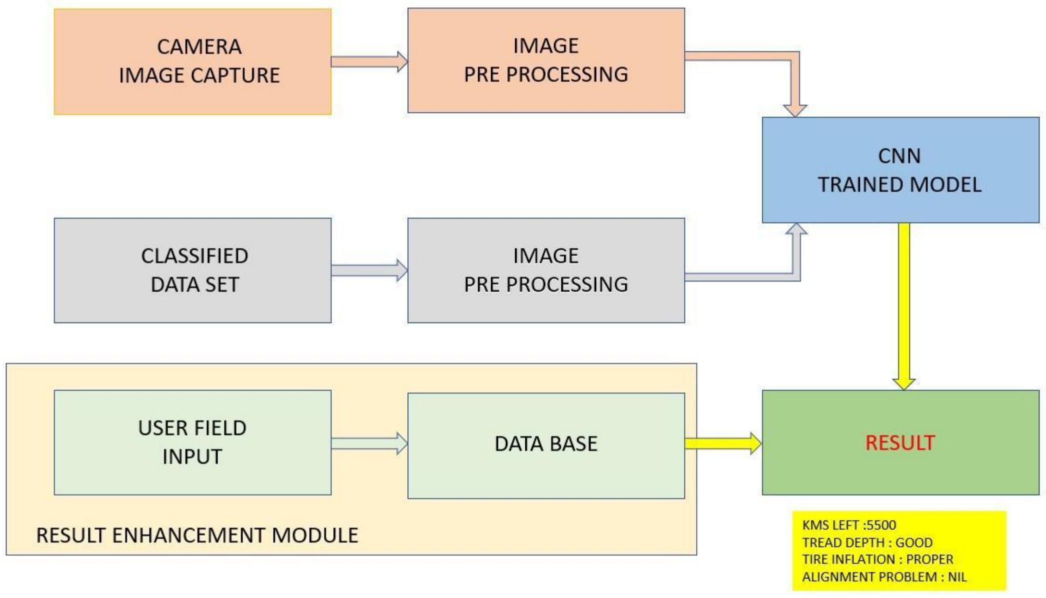


Fig 3.2: Block Diagram

REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

- Python
- Google Colab
- Pip
- Numpy
- Flask
- TensorFlow
- Keras
- Android Studio
- Laptop
- Heroku

4.2 GOOGLE COLAB

Colab is a free notebook environment that runs entirely in the cloud. Colab supports many popular machine learning libraries which can be easily loaded in your notebook. Another attractive feature that Google offers to developers is the use of GPU. Colab supports GPU and it is free.

4.3 TENSORFLOW

TensorFlow is an end-to-end open-source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use low-level APIs to create and explore new machine learning algorithms.

4.4 FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools.

4.5 HEROKU

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and similarly scale applications across most languages.

CLASSIFICATION MODEL & ALGORITHM DESIGN

5.1 CLASSIFICATION MODEL

The classification model is, in some ways, the simplest of the several types of predictive analytics models we’re going to cover. It puts data in categories based on what it learns from historical data.

5.2 CONVOLUTIONAL NEURAL NETWORK(CNN)

When it comes to Machine Learning, Artificial Neural Networks perform well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with

learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which is used to minimize the loss.

Convolutional Neural Networks or covers are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as images generally have red, green, and blue channels).

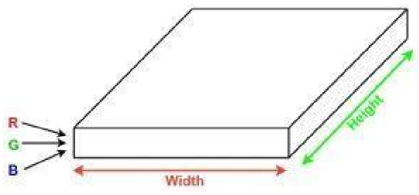


Fig 5.1: Image Representation with red green and blue channel

Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs, and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. his operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension 34x34x3. The possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

Types of layers:

Let's take an example by running a covnets on an image of dimension 32 x 32 x 3.

1. **Input Layer:** This layer holds the raw input of the image with width 32, height 32, and depth 3.
2. **Convolution Layer:** This layer computes the output volume by the computing dot product between all filters and image patch. Suppose we use a total of 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.
3. **Activation Function Layer:** This layer will apply an element-wise activation function to the output of the convolution layer. Some common activation

functions are RELU: max(0, x), Sigmoid: 1/(1+e^-x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.

4. **Pool Layer:** This layer is periodically inserted in the comments and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

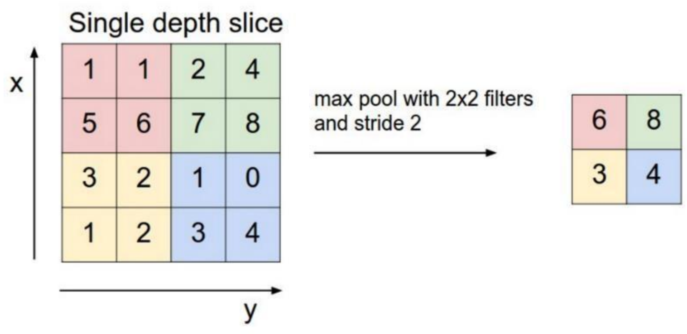


Fig 5.2: Pool Layer

5. **Fully-Connected Layer:** This layer is a regular neural network layer that takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

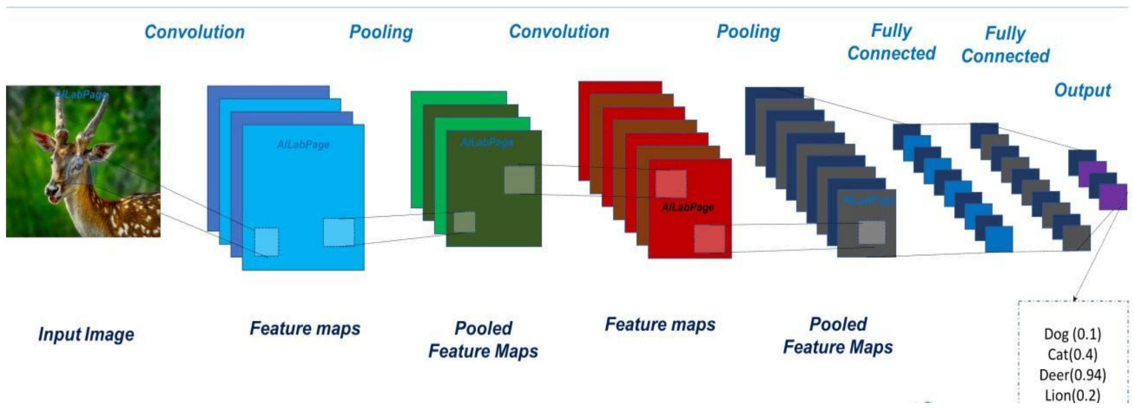


Fig 5.3: Convolutional Neural Network Layers

5.3 TYRE LIFE PREDICTION ALGORITHM

The Tyre defect will be detected by the CNN model. But the model does not give the life prediction that this particular tyre can last a few thousand kilometers. For this, We have designed an algorithm that can get the inputs from the user through a mobile app. These inputs are the one which cannot be extracted from the tyre image. The algorithm includes all the crucial parameters that affect the tyre life both mechanical and physical. With the inputs from the user and the result from the prediction model our system can provide users with the result, data contains the feature use of the tyre and also recommendations based on the prediction results to improve the tyre life further.

Manufacture Claim Life	Kms used till now	remaining	Air Pressure		Alignment Problem		Usage Area			Average Speed	
			<15 days	>15days	<=5000 kms	>5000kms	Highway	mixed	Broken	<70 kmph	> 70kmph
loss in life percentage			2%	10%	3%	10%	5%	10%	15%	3%	10%
40000	20000	20000	400	2000	600	2000	1000	2000	3000	600	1000

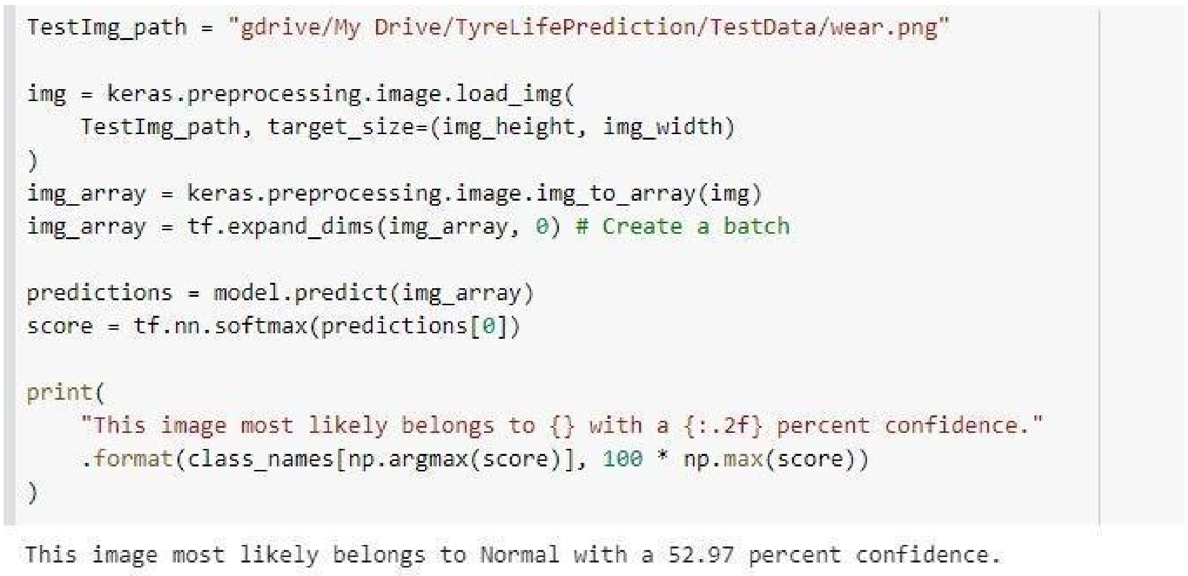
Fig 5.4: Prediction Table

🟩	Worst Case
🟨	Ideal Case

RESULT ANALYSIS

We Tested our system with a variety of test data and the system is found to be capable of classifying the test data with greater accuracy and life prediction results are also accurate.

6.1 DEFECT DETECTION

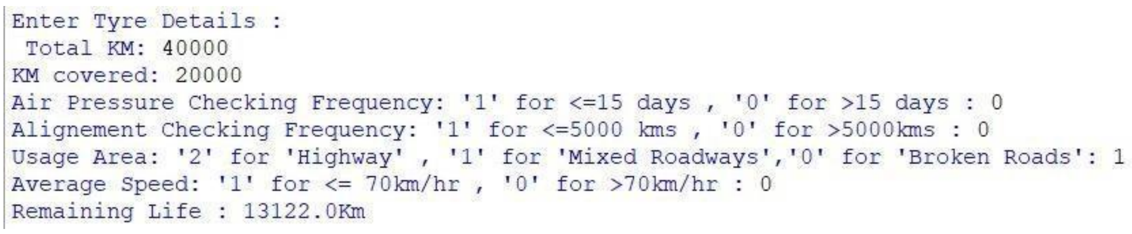


```
TestImg_path = "gdrive/My Drive/TyreLifePrediction/TestData/wear.png"

img = keras.preprocessing.image.load_img(
    TestImg_path, target_size=(img_height, img_width)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to Normal with a 52.97 percent confidence.
```

Fig 6.1: Defect Detection

6.2 LIFE PREDICTION



```
Enter Tyre Details :
 Total KM: 40000
KM covered: 20000
Air Pressure Checking Frequency: '1' for <=15 days , '0' for >15 days : 0
Alignement Checking Frequency: '1' for <=5000 kms , '0' for >5000kms : 0
Usage Area: '2' for 'Highway' , '1' for 'Mixed Roadways','0' for 'Broken Roads': 1
Average Speed: '1' for <= 70km/hr , '0' for >70km/hr : 0
Remaining Life : 13122.0Km
```

Fig 6.2: Life Prediction

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

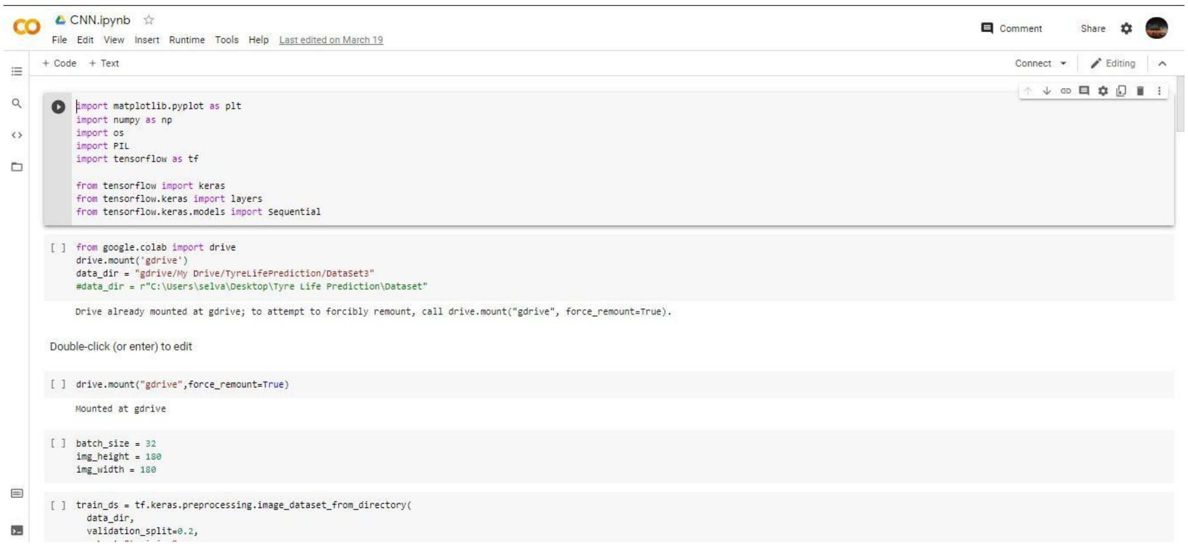
The tyre life prediction system can find the Bulges, Sidewall cracking, Air Inflation, Alignment issues, and Treadwear. Pretrained model is used in our system to save computation time and resources. The automated inspection method will allow the vehicle users to monitor their tyre life with their own hands instead of getting the vehicle to the garage spending time to diagnose it. With the image of the thread section of the tyre, the real-time condition of tyre is predicted. Then, the image will be processed to extract the thread profile and tread depth from the image. The depth calculation method to be improved. The future prediction system would include a lot more parameters and also result would be more informative by including analysis-based ratings and recommendations to improve the current tyres life to get the maximum out of tyre. A lot more types of tyre models to be included and also a more precise algorithm to be developed for the retreaded tyres and commercial vehicle tyres by taking their route of operation and load conditions into account. Another image of the sidewall of the same tyre will be captured, which gives details such as tyre brand, model, and size. The depth will be calculated and information will be processed to provide the prediction output in Kilometers. Tyre plays a key role in vehicle safety, resolving such types of issues we need such tools and techniques end of the day such systems will boost the confidence while drive and saves significant money by extracting the maximum out of the tyres.

7.2 FUTURE SCOPE

In the future, the system will be further tuned to predict the life of the bias-ply tyre and rethread tyre more efficiently. And also instead of selecting the tyre model from the drop-down in the android app, the system will be developed to read the tyre data from the sidewall image. Now the system has three major categories of defects in the future it will be more divided into more categories to cover all types of major and minor defective parameters both physical and mechanical

APPENDIX

GOOGLE COLLAB


CNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 19
Comment Share
+ Code + Text
Connect Editing

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
from google.colab import drive
drive.mount('gdrive')
data_dir = "gdrive/My Drive/TyreLifePrediction/DataSet3"
#data_dir = r"C:\Users\selva\Desktop\Tyre Life Prediction\Dataset"
```

```
Drive already mounted at gdrive; to attempt to forcibly remount, call drive.mount("gdrive", force_remount=True).
```

Double-click (or enter) to edit

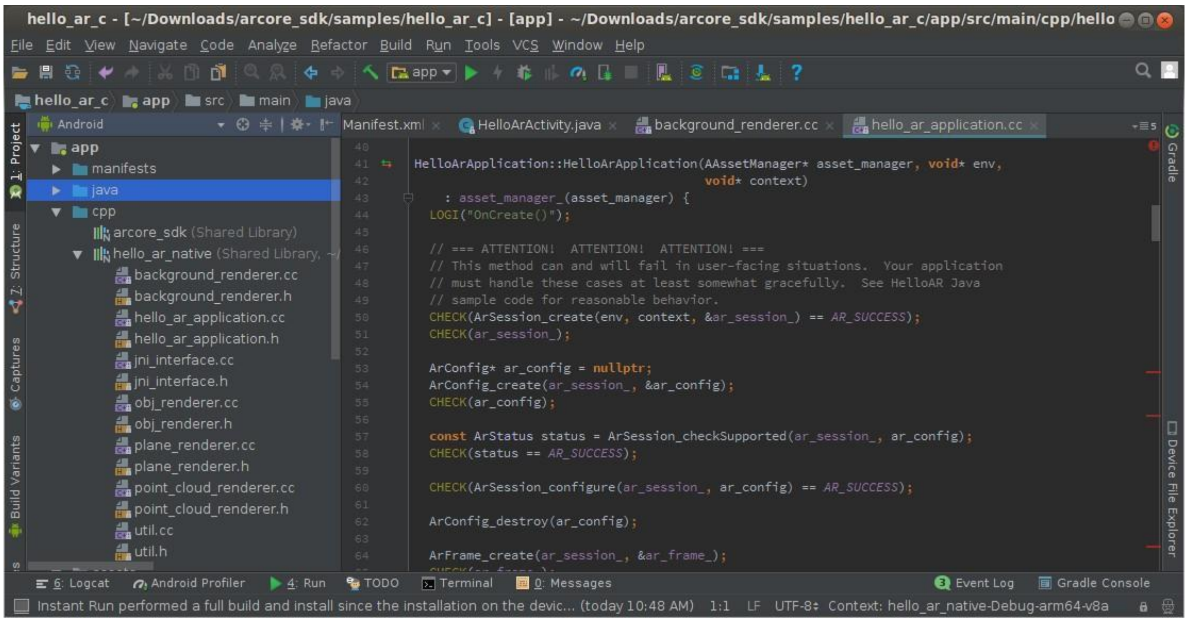
```
drive.mount("gdrive",force_remount=True)
```

```
Mounted at gdrive
```

```
batch_size = 32
img_height = 180
img_width = 180
```

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
```

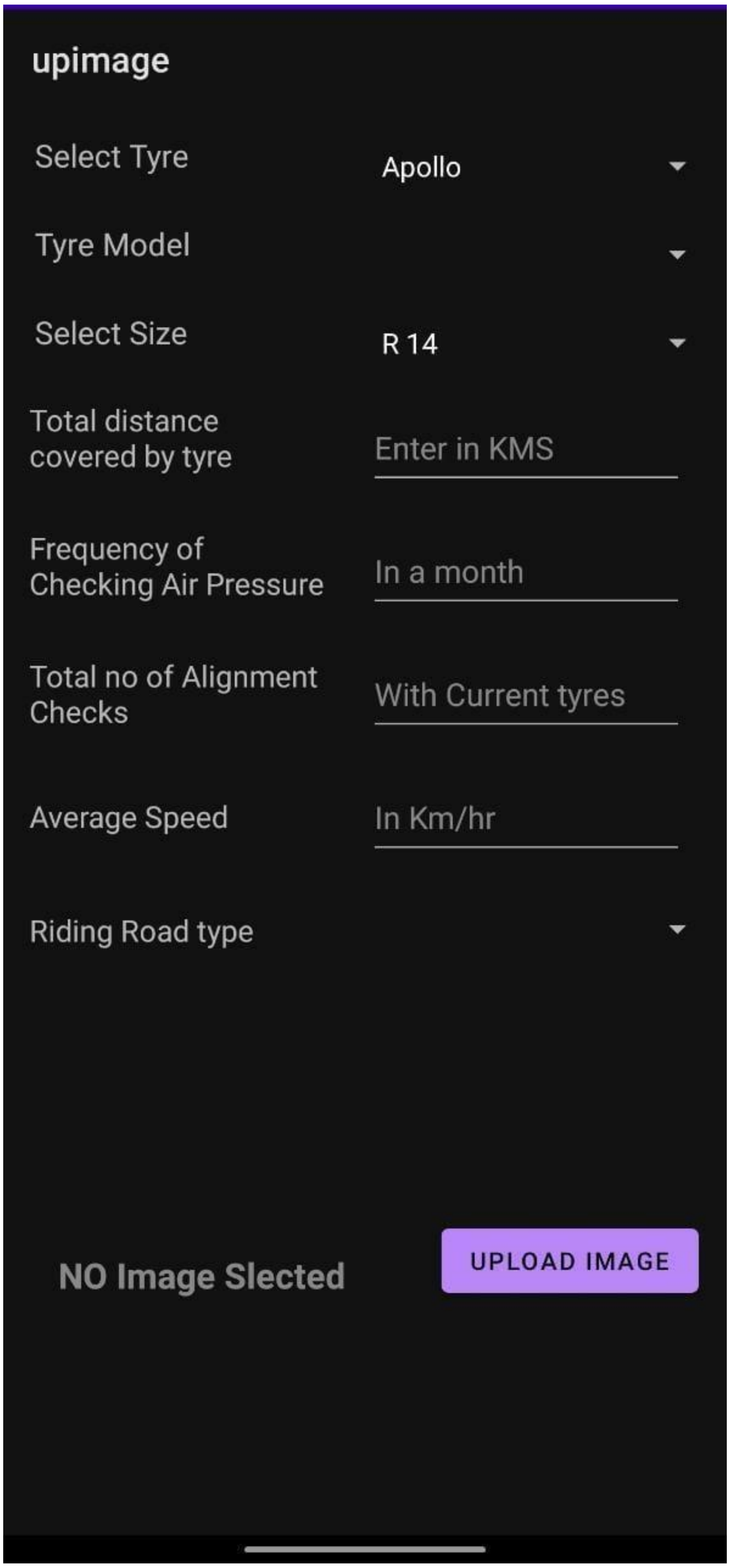
ANDROID STUDIO



```
HelloArApplication::HelloArApplication(AAssetManager* asset_manager, void* env,
                                       void* context)
    : asset_manager_(asset_manager) {
  LOGI("OnCreate()");

  // === ATTENTION!  ATTENTION!  ATTENTION! ===
  // This method can and will fail in user-facing situations.  Your application
  // must handle these cases at least somewhat gracefully.  See HelloAR Java
  // sample code for reasonable behavior.
  CHECK(ArSession_create(env, context, &ar_session_) == AR_SUCCESS);
  CHECK(ar_session_);

  ArConfig* ar_config = nullptr;
  ArConfig_create(ar_session_, &ar_config);
  CHECK(ar_config);

  const ArStatus status = ArSession_checkSupported(ar_session_, ar_config);
  CHECK(status == AR_SUCCESS);

  CHECK(ArSession_configure(ar_session_, ar_config) == AR_SUCCESS);

  ArConfig_destroy(ar_config);

  ArFrame_create(ar_session_, &ar_frame_);
```

```
Instant Run performed a full build and install since the installation on the devic... (today 10:48 AM)   1:1   LF   UTF-8   Context: hello_ar_native-Debug-arm64-v8a
```

ANDROID APP


upimage

Select Tyre	Apollo
Tyre Model	
Select Size	R 14
Total distance covered by tyre	Enter in KMS
Frequency of Checking Air Pressure	In a month
Total no of Alignment Checks	With Current tyres
Average Speed	In Km/hr
Riding Road type	

NO Image Slected

SOURCE CODE

```
  import matplotlib.pyplot as plt import
numpy as np import os import PIL import
tensorflow as tf

  from tensorflow import keras from tensorflow.keras import
layers from tensorflow.keras.models import Sequential

  from google.colab import drive drive.mount('gdrive') data_dir = "gdrive/My
Drive/TyreLifePrediction/DataSet3" #data_dir = r"C:\Users\selva\Desktop\Tyre
Life Prediction\Dataset"

  drive.mount("gdrive",force_remount=True)

  batch_size = 32 img_height = 180
img_width = 180

  train_ds = tf.keras.preprocessing.image_dataset_from_directory(   data_dir,
validation_split=0.2,   subset="training",   seed=123,
image_size=(img_height, img_width),
    batch_size=batch_size)

  val_ds = tf.keras.preprocessing.image_dataset_from_directory(   data_dir,
validation_split=0.2,   subset="validation",   seed=123,
image_size=(img_height, img_width),   batch_size=batch_size)

  class_names = train_ds.class_names print(class_names)
  import matplotlib.pyplot as plt

  plt.figure(figsize=(10, 10)) for images, labels in
train_ds.take(1):   for i in range(9):
      ax = plt.subplot(3, 3, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(class_names[labels[i]])     plt.axis("off")

  for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
print(labels_batch.shape)   break

  AUTOTUNE = tf.data.AUTOTUNE

  train_ds   =   train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

  normalization_layer = layers.experimental.preprocessing.Rescaling(1./255)

  normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds)) first_image =
image_batch[0]
  # Notice the pixels values are now in `[0,1]`. print(np.min(first_image),
np.max(first_image))

  num_classes = len(class_names)

  model = Sequential([
layers.experimental.preprocessing.Rescaling(1./255,
input_shape=(img_height, img_width, 3)),
layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),   layers.Conv2D(32, 3,
padding='same', activation='relu'),   layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),   layers.Flatten(),
layers.Dense(128, activation='relu'),
layers.Dense(num_classes)
  ])

  model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])

  model.summary()

  epochs=10 history =
model.fit(   train_ds,
validation_data=val_ds,
epochs=epochs
  )

  TestImg_path = "gdrive/My Drive/TyreLifePrediction/TestData/wear.png"

  model = Sequential([   data_augmentation,

layers.experimental.preprocessing.Rescaling(1./255),

layers.Conv2D(16, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),   layers.Conv2D(32, 3,
padding='same', activation='relu'),   layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),   layers.Dropout(0.2),
   layers.Flatten(),   layers.Dense(128,
activation='relu'),   layers.Dense(num_classes)
 ])


 model.compile(optimizer='adam',


 loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])


 model.summary()


 epochs = 15 history = model.fit(
train_ds,   validation_data=val_ds,
epochs=epochs
 )
```

```
  acc = history.history['accuracy'] val_acc =
history.history['val_accuracy'] loss =
history.history['loss'] val_loss =
history.history['val_loss']

  epochs_range = range(epochs)

  plt.figure(figsize=(8, 8)) plt.subplot(1, 2, 1) plt.plot(epochs_range, acc,
label='Training Accuracy') plt.plot(epochs_range, val_acc, label='Validation
Accuracy') plt.legend(loc='lower right') plt.title('Training and Validation
Accuracy')

  plt.subplot(1, 2, 2) plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right') plt.title('Training and Validation Loss')
plt.show()

  TestImg_path = "gdrive/My
  Drive/TyreLifePrediction/TestData/TyreBuldge.jpg"

  img = keras.preprocessing.image.load_img(
     TestImg_path, target_size=(img_height, img_width)
  )
  img_array = keras.preprocessing.image.img_to_array(img) img_array =
tf.expand_dims(img_array, 0) # Create a batch


  predictions = model.predict(img_array) score =
tf.nn.softmax(predictions[0])


  print(
     "This image most likely belongs to {} with a {:.2f} percent confidence."
     .format(class_names[np.argmax(score)], 100 * np.max(score))
```

ANDROID APP SORCE CODE

```
 kage com.example.uploadimg; import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.app.ActivityCompat; import
androidx.core.content.ContextCompat;

 import android.Manifest; import android.app.ProgressDialog; import
android.content.Intent; import android.content.pm.PackageManager; import
android.database.Cursor; import android.graphics.Bitmap; import
android.net.Uri; import android.os.Bundle; import android.provider.MediaStore;
```

```
import android.util.Log; import android.view.View; import
android.widget.AdapterView; import android.widget.ArrayAdapter; import
android.widget.Button; import android.widget.EditText; import
android.widget.ImageView; import android.widget.Spinner; import
android.widget.TextView; import android.widget.Toast;

 import com.android.volley.AuthFailureError; import
com.android.volley.NetworkResponse; import com.android.volley.Request;
import com.android.volley.RequestQueue; import
com.android.volley.Response; import com.android.volley.VolleyError; import
com.android.volley.toolbox.Volley; import org.json.JSONException; import
org.json.JSONObject; import java.io.ByteArrayOutputStream; import
java.io.IOException; import java.util.HashMap; import java.util.Map; public
class MainActivity extends AppCompatActivity {
    Spinner classSpinner, divSpinner;
            String   selectedClass,   selectedDiv,selectRoad,selectsize;           int
dis,psi,align,speed;
     String sdis,spsi,salign,sspeed,stdis,r;
     EditText DistanceInput;
     EditText PressureInput;
     EditText AlignmentInput;
    EditText SpeedInput;

    Button upload;

    ProgressDialog progressDialog;

    private static final String ROOT_URL =
"https://tlps.herokuapp.com/tyreLife";     private static final int
REQUEST_PERMISSIONS = 100;     private static final int
PICK_IMAGE_REQUEST =1 ;     private Bitmap bitmap;     private String
filePath;

    ImageView imageView;

    TextView textView;


    @Override     protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);


                //initializing    views                    imageView    =
findViewById(R.id.imageView);                                textView    =
findViewById(R.id.textview);                  DistanceInput   =   (EditText)
findViewById(R.id.DistanceInput);

      PressureInput = (EditText) findViewById(R.id.PressureInput);
```

```
        AlignmentInput = (EditText) findViewById(R.id.AlignmentInput);
SpeedInput = (EditText) findViewById(R.id.SpeedInput);        classSpinner =
(Spinner) findViewById(R.id.classSpinner);        divSpinner = (Spinner)
findViewById kage com.example.uploadimg; import
androidx.appcompat.app.AppCompatActivity; import
androidx.core.app.ActivityCompat; import
androidx.core.content.ContextCompat;


  import android.Manifest; import
android.app.ProgressDialog; import
android.content.Intent; import
android.content.pm.PackageManager; import
android.database.Cursor; import
android.graphics.Bitmap; import android.net.Uri;
import android.os.Bundle; import
android.provider.MediaStore; import
android.util.Log;
  import android.view.View; import android.widget.AdapterView;
import android.widget.ArrayAdapter; import android.widget.Button;
import android.widget.EditText; import android.widget.ImageView;
import android.widget.Spinner; import android.widget.TextView;

import android.widget.Toast;


  import com.android.volley.AuthFailureError; import

com.android.volley.NetworkResponse; import com.android.volley.Request;

import com.android.volley.RequestQueue; import com.android.volley.Response;

import com.android.volley.VolleyError; import

com.android.volley.toolbox.Volley;


  import org.json.JSONException; import org.json.JSONObject; import

java.io.ByteArrayOutputStream; import java.io.IOException; import

java.util.HashMap; import java.util.Map;


  public class MainActivity extends AppCompatActivity {



     Spinner classSpinner, divSpinner;

             String    selectedClass,    selectedDiv,selectRoad,selectsize;                int

dis,psi,align,speed;

      String sdis,spsi,salign,sspeed,stdis,r;
```

```
    EditText DistanceInput;

    EditText PressureInput;

    EditText AlignmentInput;

    EditText SpeedInput;


    Button upload;

    ProgressDialog progressDialog;


    private static final String ROOT_URL =
"https://tlps.herokuapp.com/tyreLife";     private static final int
REQUEST_PERMISSIONS = 100;     private static final int
PICK_IMAGE_REQUEST =1 ;     private Bitmap bitmap;     private String
filePath;

    ImageView imageView;

    TextView textView;


    @Override     protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

```
                    //initializing    views                         imageView   =
findViewById(R.id.imageView);                                  textView    =
findViewById(R.id.textview);                   DistanceInput  =  (EditText)
findViewById(R.id.DistanceInput);

        PressureInput = (EditText) findViewById(R.id.PressureInput);
                AlignmentInput  =  (EditText)  findViewById(R.id.AlignmentInput);
SpeedInput = (EditText) findViewById(R.id.SpeedInput);


                    classSpinner   =   (Spinner)   findViewById(R.id.classSpinner);
divSpinner = (Spinner) findViewById(R.id.divSpinner);
        Spinner Roade = (Spinner) findViewById(R.id.Roade);
        Spinner tsize = (Spinner) findViewById(R.id.tsize); (R.id.divSpinner);
        Spinner Roade = (Spinner) findViewById(R.id.Roade);
        Spinner tsize = (Spinner) findViewById(R.id.tsize);
```

ANDROID MANIFEST

```
 <?xml version="1.0" encoding="UTF-8"?>
 <manifest package="com.example.uploadimg"
xmlns:android="http://schemas.android.com/apk/res/android"><usespermission
android:name="android.permission.INTERNET"/><usespermission
 android:name="android.permission.READ_EXTERNAL_STORAGE"/><uses
 -permission
 android:name="android.permission.WRITE_EXTERNAL_STORAGE"/><ap
 plication android:theme="@style/Theme.Uploadimg"
android:supportsRtl="true"
 android:roundIcon="@mipmap/ic_launcher_round"
 android:label="@string/app_name" android:icon="@mipmap/ic_launcher"
android:allowBackup="true"><activity
android:name=".MainActivity"><intent-filter><action
android:name="android.intent.action.MAIN"/><category
android:name="android.intent.category.LAUNCHER"/></intentfilter></activity
></application></manifest>
```

HEROKU SERVER

DEMO VIDEO

https://vimeo.com/564669980