MIRACLE
SOFTWARE SYSTEMS

Building a Car Damage Classification model using Google Colab

Workshop | Digital Summit 2019

Building a Car Damage Classification model using Google Colab

Introduction

This document contains a step-by-step process for creating a Machine Learning model in Google Colab. We will teach you the end to end process of building a ML model to predict the part of the car that is damaged.

This guide was prepared by **Miracle's Innovation Labs**.

Pre-Requisites

All attendees must have their workstation (with Internet) to participate in the lab. The following prerequisites will help you to make the Hands-on Lab experience easier.

Technology Involved

- Python 3
- Keras
- Google Colab

Lab Steps

So, let us get started with the model!

The following steps will be an outline on how you can create a Machine Learning model using Google Colab. This model helps in predicting the damaged part of a car based on the dataset containing the damaged cars.

Step #1 | Creation of Python Notebook in Google Colab

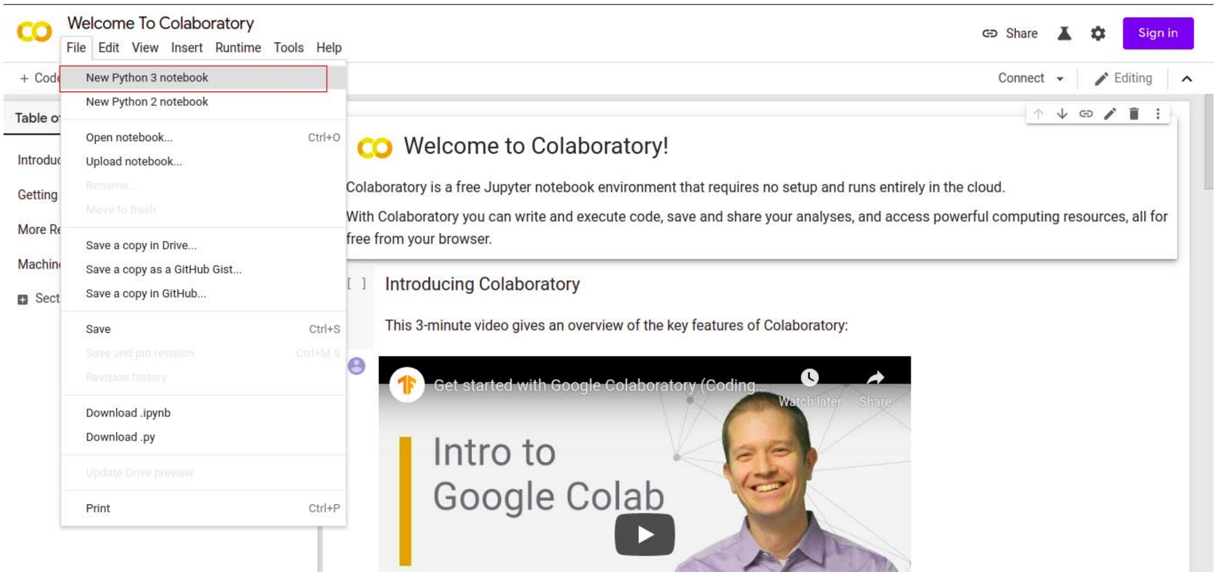
Google Colab is an IDE to run interactive python scripts. The local version of Google Colab is Jupyter Notebook.

Following are the steps to set an environment for building a Deep Learning Model.

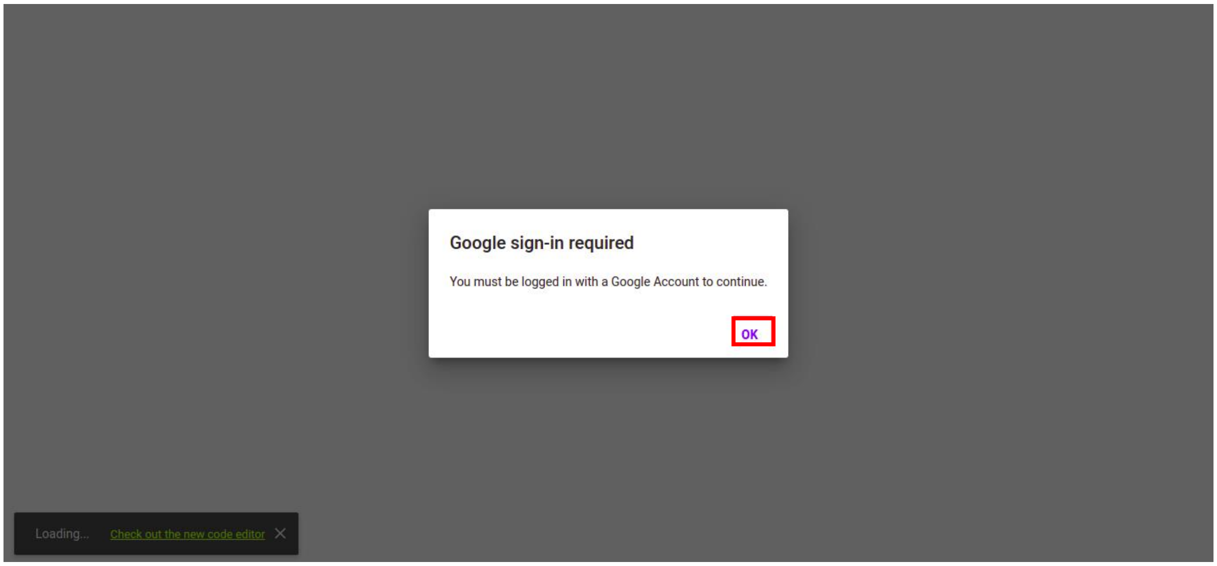
Below is the link for Google Colab home page,

https://colab.research.google.com/notebooks/welcome.ipynb

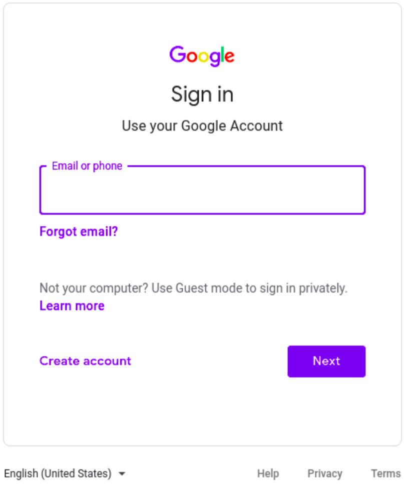
Click on File → New Python3 notebook



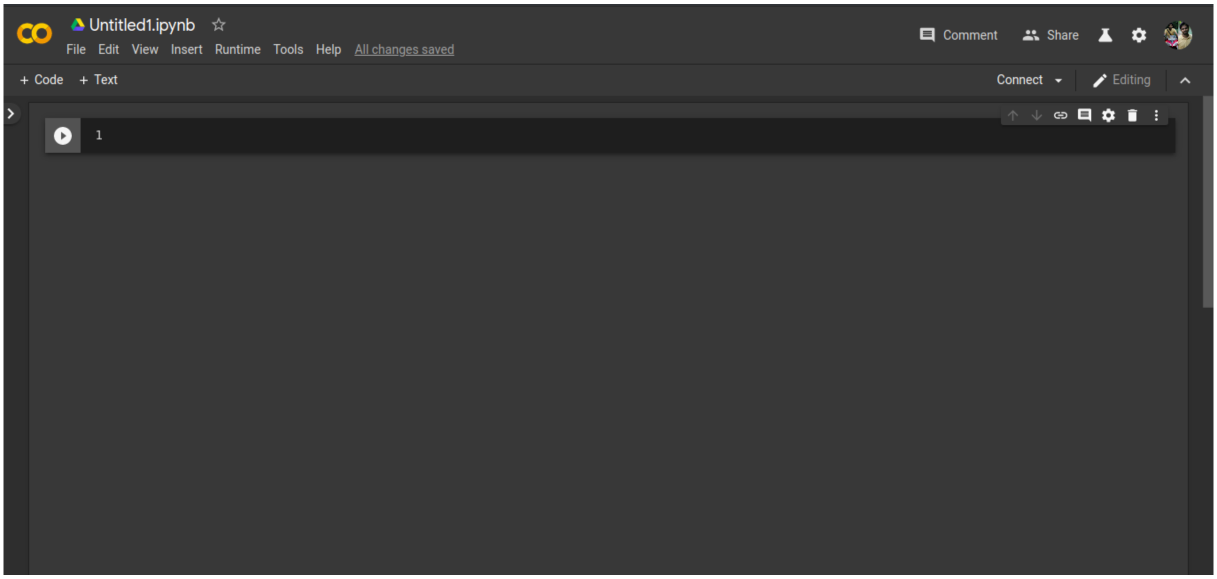
You will be redirected to a webpage where an alert box is displayed titled as **Google sign-in required**. Click **OK**



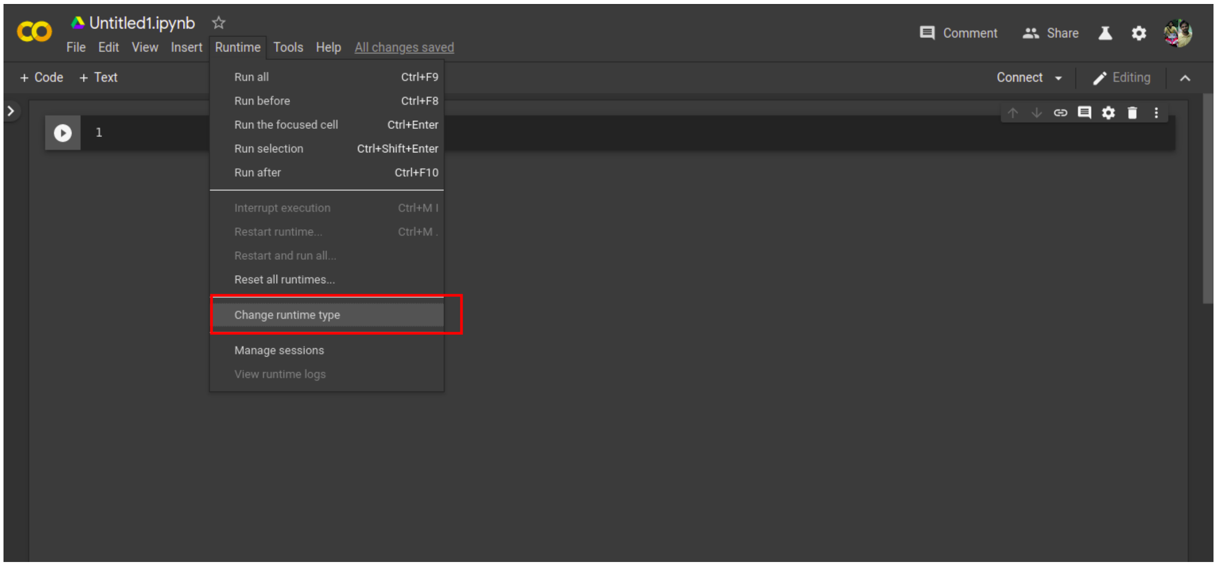
Sign in to your Google Account



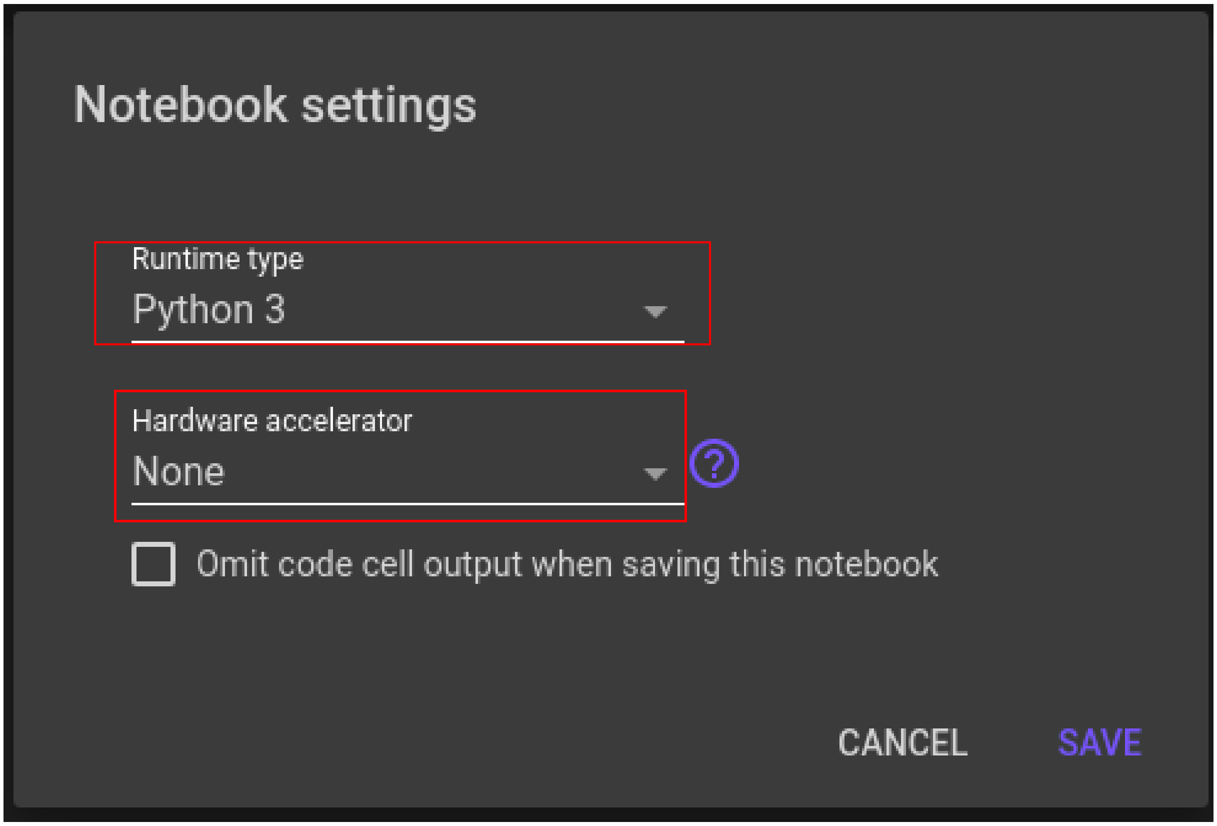
After signing in, a new Python3 notebook is opened named as **Untitled1.ipynb**. Rename it by clicking on the file name with extension as **.ipynb**



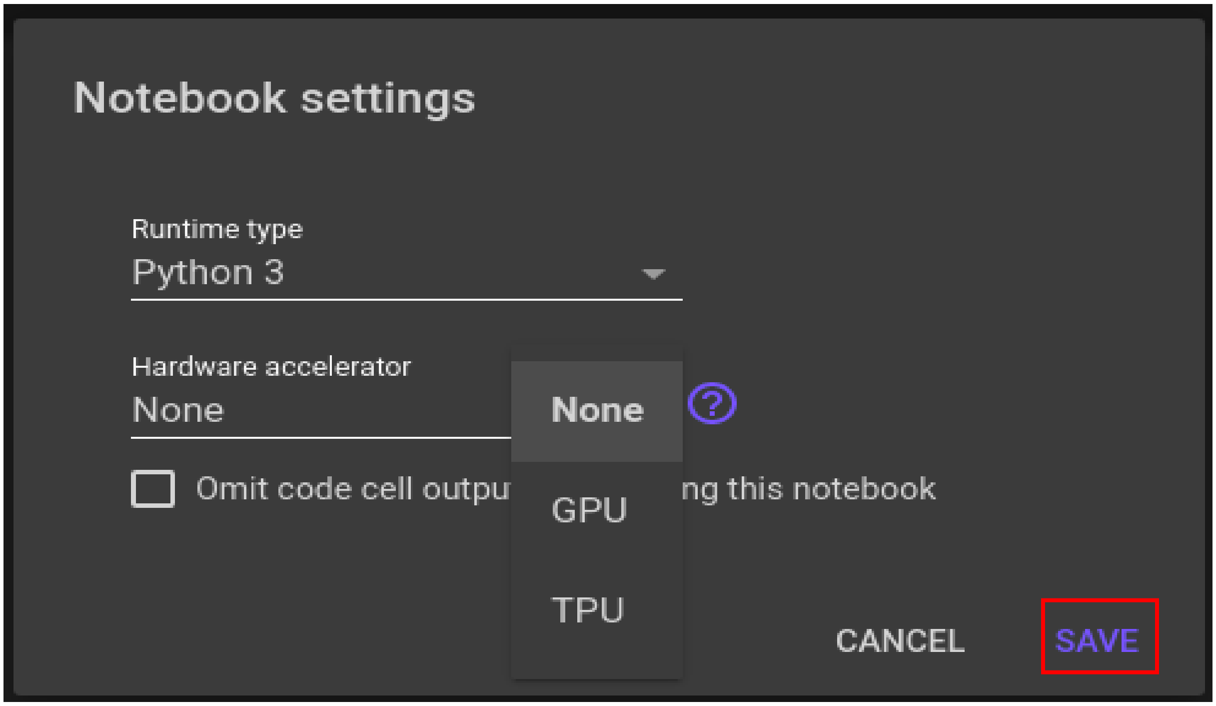
Change the runtime to **GPU** by clicking on **Runtime → Change runtime Type**



A dialog box titled **Notebook settings** appears



Change the **Hardware accelerator** to **GPU** and click on **SAVE**



Step #2 | Dataset Collection

The first step is collecting the dataset. Dataset is nothing but a folder of images which is used for training the model. Since there is no standard dataset available for damaged car images, the images are downloaded from Google.

The file structure must be,

- Training
  - Bumper
  - Door
  - Glass
- Testing
  - Bumper
  - Door
  - Glass

Note: We need to make sure that at least 400 images are downloaded for each class.

Step #3 | Access the Dataset from GitHub

Clone the dataset from GitHub for training the model in Google Colab.

```
!git clone https://github.com/ammu11/DS19-DamageCarClassification
```

When the dataset is cloned successfully from the GitHub it shows as below,

```
Cloning into 'DS19-DamageCarClassification'...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 1787 (delta 3), reused 17 (delta 1), pack-reused 1768
Receiving objects: 100% (1787/1787), 133.63 MiB | 27.06 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Step #4 | Model Building

Image classification can be done using CNN. And also Keras have various pre-trained models similar to CNN like **InceptionV3**, **InceptionResNetV2**, **Mobile Net**, etc. For more information refer https://keras.io/applications/

Following are the steps of model building,

Clone the Git repo which has the dataset.

Import required libraries

```
import keras
import os
from keras.callbacks import ModelCheckpoint
from keras import backend as K
from keras.layers.core import Dense, Activation
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.models import Model
from keras.applications import imagenet_utils
from keras import regularizers
from keras.layers import Dense,GlobalAveragePooling2D, Dropout
from keras.applications import MobileNet
from keras.applications.mobilenet import preprocess_input
import numpy as np
from IPython.display import Image
from keras.optimizers import Adam
import os.path as osp
import argparse
import tensorflow as tf
from keras.models import load_model
```

Declare the test and train folder paths

```
train_dir = "/content/DS19-DamageCarClassification/training"
test_dir = "/content/DS19-DamageCarClassification/testing"
```

Model generation

Consider the model

```
base_model=MobileNet(weights='imagenet',include_top=False)
x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu', kernel_regularizer=regularizers.l2(0.001))(x)
x=Dense(512,activation='relu', kernel_regularizer=regularizers.l2(0.001))(x)
preds=Dense(3,activation='softmax', kernel_regularizer=regularizers.l2(0.001))(x)
```

Once the layers are created, the result will be as follows,

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is dep

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecat

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is depr

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecat

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is deprecated.

/usr/local/lib/python3.6/dist-packages/keras_applications/mobilenet.py:207: UserWarning: `input_shape` is undefined or non-square, or `rows
  warnings.warn('`input_shape` is undefined or non-square, '
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is dep

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer i

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_defaul

Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.6/mobilenet_1_0_224_tf_no_top.h5
17227776/17225924 [==============================] - 1s 0us/step
```

Specify the model with respective input and output parameters

```
model=Model(inputs=base_model.input,outputs=preds)
```

Fine tuning the model

```
for layer in model.layers[:20]:
  layer.trainable=False
for layer in model.layers[20:]:
  layer.trainable=True
```


Data augmentation

```
img_width, img_height = 224, 224
epochs = 200
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
validation_generator = train_datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
```

When data augmentation is done, it gives the number of images for total number of classes

```
Found 1366 images belonging to 3 classes.
Found 336 images belonging to 3 classes.
```

Creation of checkpoints

```
savepath = os.path.join( ""+ 'checkpoint-{epoch:03d}.h5' )
checkpointer = ModelCheckpoint(filepath=savepath,monitor='val_acc', mode='max', verbose=0, save_best_only=True)
```

Train the model

```
model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit_generator(
        train_generator,
        steps_per_epoch=train_generator.samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=validation_generator.samples // batch_size,
        callbacks=[checkpointer])
```

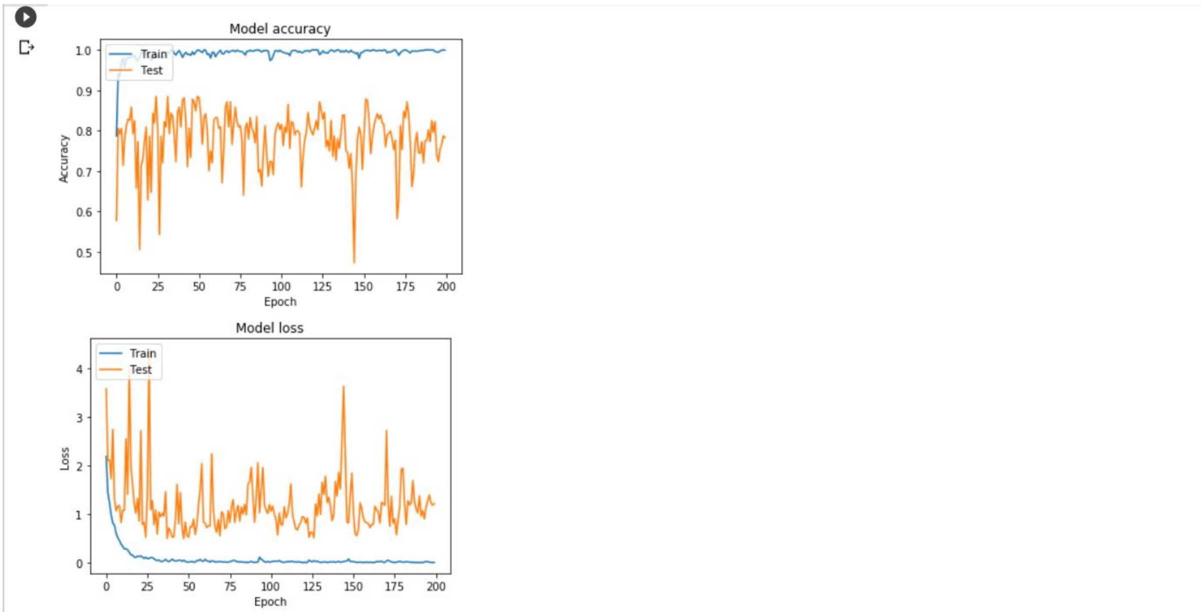
The model is fitted against all the parameters passed and there by model training is started

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is deprecated. Pl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.pythor
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is depreca

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated.

Epoch 1/200
18/42 [===========>.................] - ETA: 16s - loss: 2.6544 - acc: 0.6679/usr/local/lib/python3.6/dist-packages/PIL/Image.py:914: Use
  'to RGBA images')
42/42 [==============================] - 18s 428ms/step - loss: 2.1909 - acc: 0.7870 - val_loss: 3.5860 - val_acc: 0.5781
Epoch 2/200
42/42 [==============================] - 12s 297ms/step - loss: 1.4499 - acc: 0.9416 - val_loss: 2.1097 - val_acc: 0.8059
```

Plot the accuracy and loss of the model

```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

The resultant graph would be as follows,



Save the model with a filename

```
model.save_weights("model.h5")
```

Access the checkpoint

```
saved_model="/content/DS19-DamageCarClassification/path_to_the_last_checkpoint_file.h5"
# (or)
#Access the last checkpoint file as final model
saved_model="/content/checkpoint-025.h5"
```

Test the model

```
def predict(model, img):
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    preds = model.predict(x)
    return preds[0]

labels = ("Bumper_Damage","Door_Damage","Glass_Damage")
model = load_model(saved_model)
img = image.load_img('/content/DS19-DamageCarClassification/testing/bumper/0048.JPEG', target_size=(224,224))
preds = predict(model, img)
j=max(preds)
result = np.where(preds == j)
index_val = result[0][0]
prediction = labels[index_val]
print("Result:",prediction)
```

Depicts the result of sample test image


```
Result: Bumper_Damage
```

For any questions regarding the lab please feel free to reach out to innovation@miraclesoft.com. **We hope you enjoyed creating Machine Learning models with us ☺**