School of Chemistry, Chemical Engineering and Biotechnology

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CH4244

Assignment 2
[Python Problem]

Name:	Lim Junwei Darien
Matric Number:	U1921811F
Date of submission:	28/11/2022

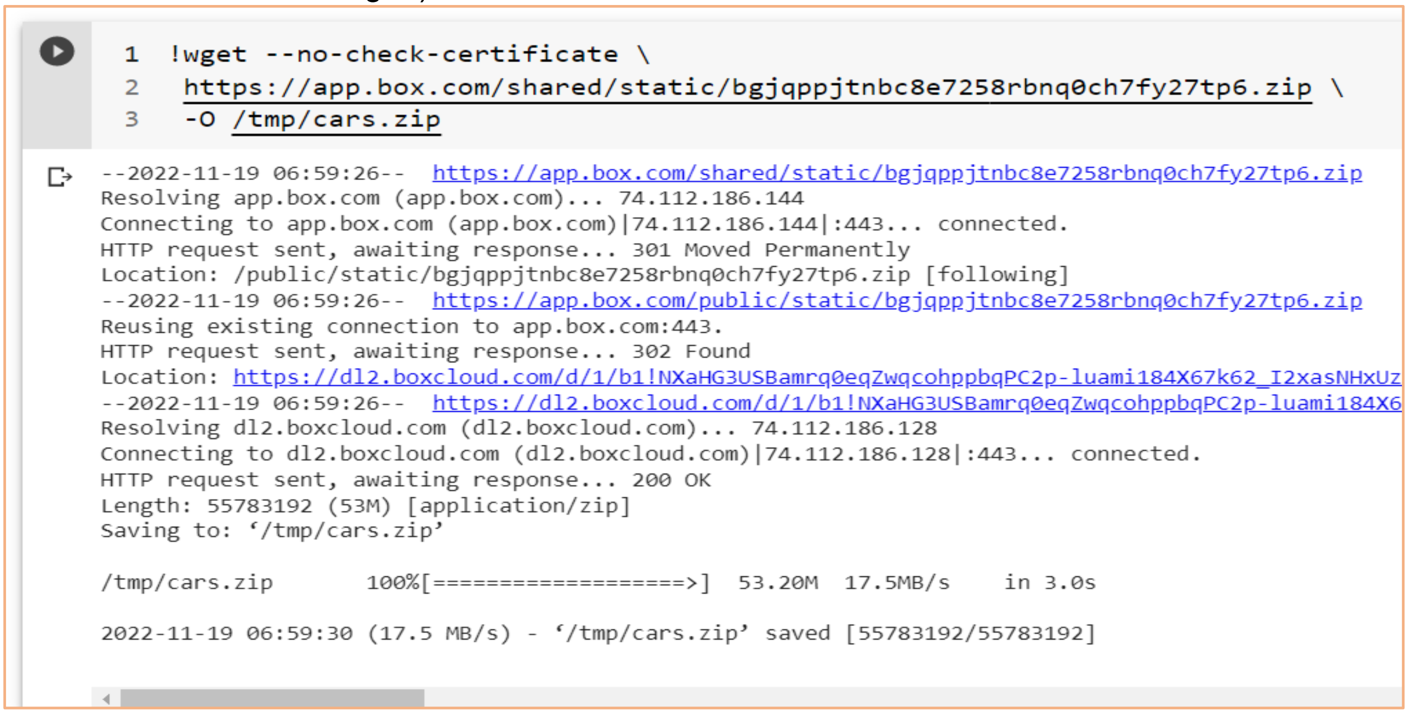
Contents

Name: Lim Junwei Darien Matriculation No.: U1921811F

a) Load the data and then preprocess them with ImageDataGenerator().

Screenshot of loaded data:

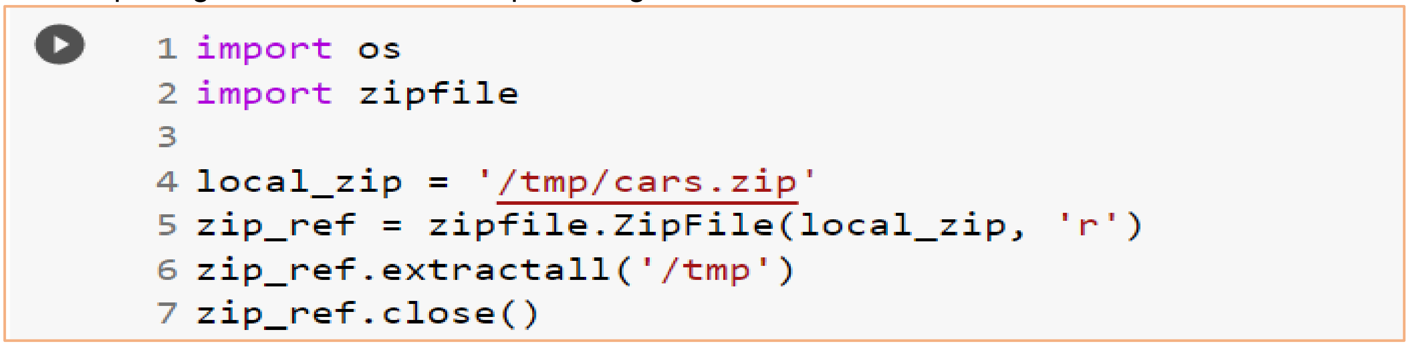
- Extract & Download data of Car Images of 5 different brands (BMW, Bugatti, Lamborghini, McLaren and Volkswagen)



```
!wget --no-check-certificate \
 https://app.box.com/shared/static/bgjqppjtnbc8e7258rbnq0ch7fy27tp6.zip \
 -O /tmp/cars.zip
```

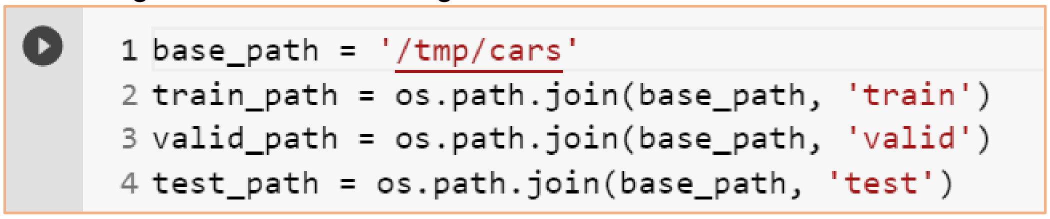
```
--2022-11-19 06:59:26--  https://app.box.com/shared/static/bgjqppjtnbc8e7258rbnq0ch7fy27tp6.zip
Resolving app.box.com (app.box.com)... 74.112.186.144
Connecting to app.box.com (app.box.com)|74.112.186.144|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /public/static/bgjqppjtnbc8e7258rbnq0ch7fy27tp6.zip [following]
--2022-11-19 06:59:26--  https://app.box.com/public/static/bgjqppjtnbc8e7258rbnq0ch7fy27tp6.zip
Reusing existing connection to app.box.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://dl2.boxcloud.com/d/1/b1!NXaHG3USBamrq0eqZwqcohppbqPC2p-luami184X67k62_I2xasNHxUz
--2022-11-19 06:59:26--  https://dl2.boxcloud.com/d/1/b1!NXaHG3USBamrq0eqZwqcohppbqPC2p-luami184X6
Resolving dl2.boxcloud.com (dl2.boxcloud.com)... 74.112.186.128
Connecting to dl2.boxcloud.com (dl2.boxcloud.com)|74.112.186.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55783192 (53M) [application/zip]
Saving to: '/tmp/cars.zip'

/tmp/cars.zip       100%[===================>]  53.20M  17.5MB/s    in 3.0s

2022-11-19 06:59:30 (17.5 MB/s) - '/tmp/cars.zip' saved [55783192/55783192]
```

- Unzip image dataset to folder.tmp and organise folder

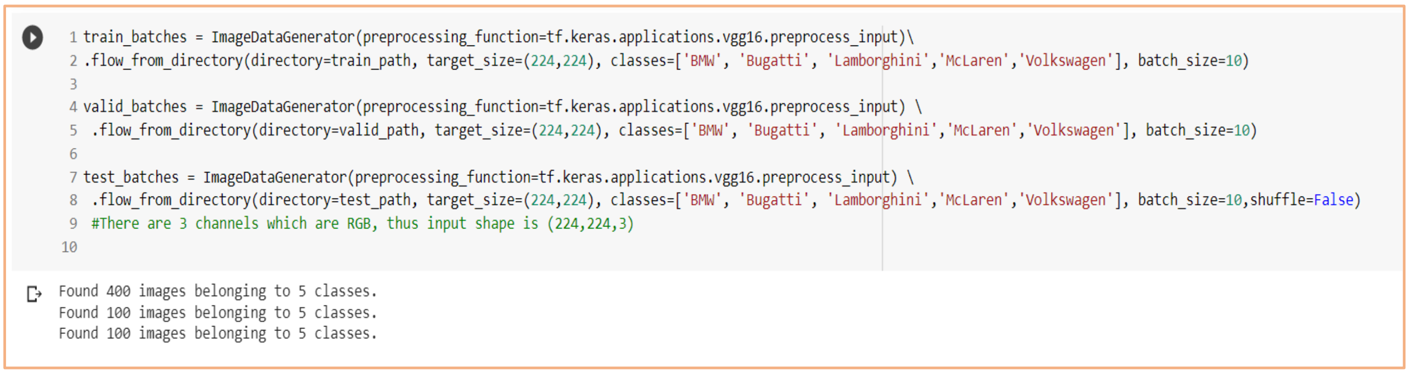


```python
import os
import zipfile

local_zip = '/tmp/cars.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp')
zip_ref.close()
```

- Putting datasets into training, validation, and test sets



```python
base_path = '/tmp/cars'
train_path = os.path.join(base_path, 'train')
valid_path = os.path.join(base_path, 'valid')
test_path = os.path.join(base_path, 'test')
```



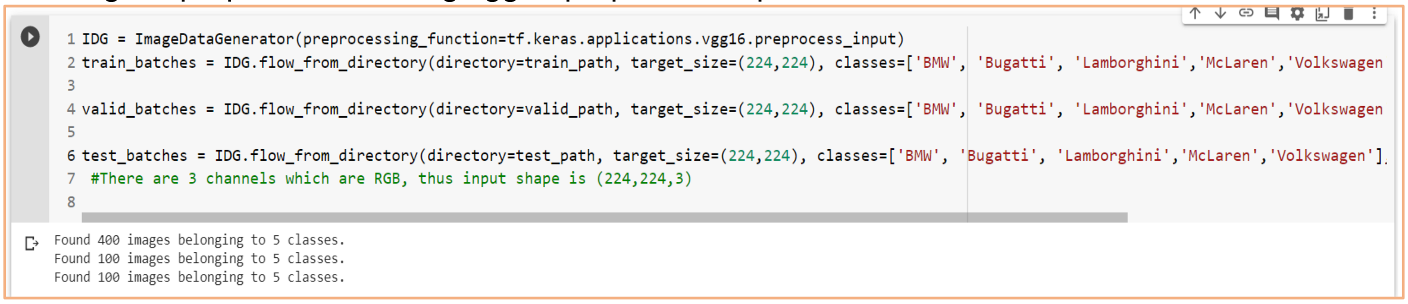
```python
train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input)\
.flow_from_directory(directory=train_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'], batch_size=10)

valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input) \
 .flow_from_directory(directory=valid_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'], batch_size=10)

test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input) \
 .flow_from_directory(directory=test_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'], batch_size=10,shuffle=False)
 #There are 3 channels which are RGB, thus input shape is (224,224,3)

```

```
Found 400 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
```

Name: Lim Junwei Darien Matriculation No.: U1921811F

Screenshot of preprocess data with ImageDataGenerator() :

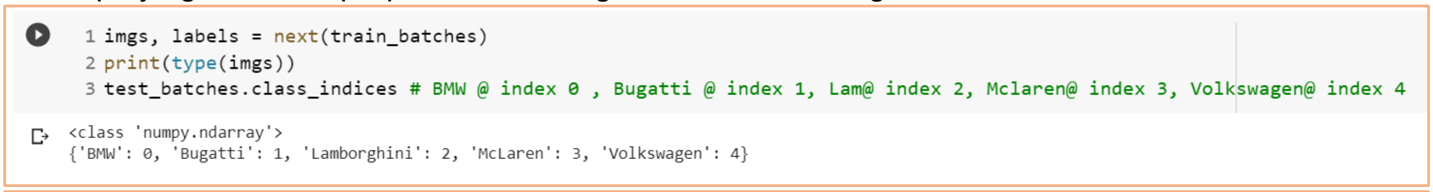
- Image is preprocessed using vgg16 preprocess input.



```
IDG = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preprocess_input)
train_batches = IDG.flow_from_directory(directory=train_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen

valid_batches = IDG.flow_from_directory(directory=valid_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen

test_batches = IDG.flow_from_directory(directory=test_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'],
 #There are 3 channels which are RGB, thus input shape is (224,224,3)

```

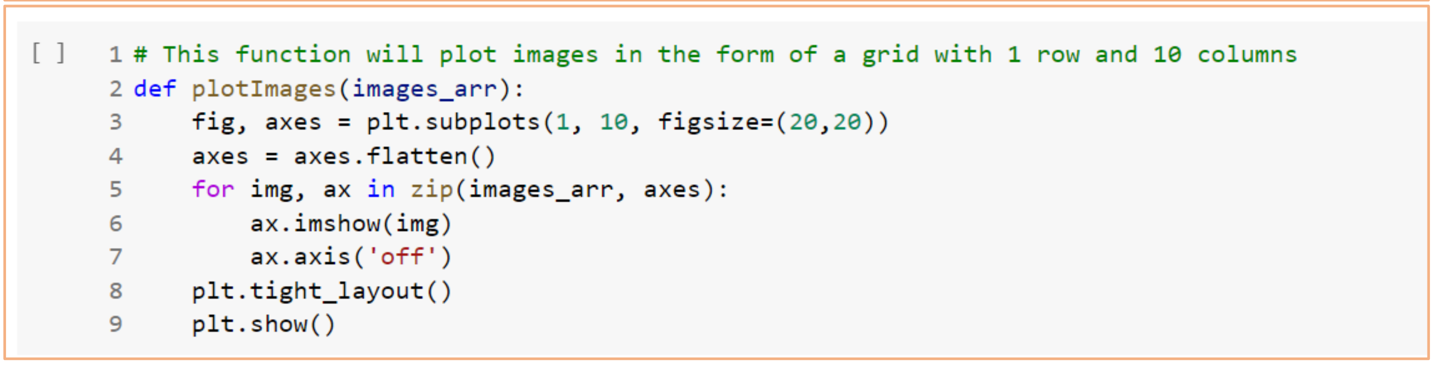
```
Found 400 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
```

- Displaying 10 of the preprocessed images from the training set

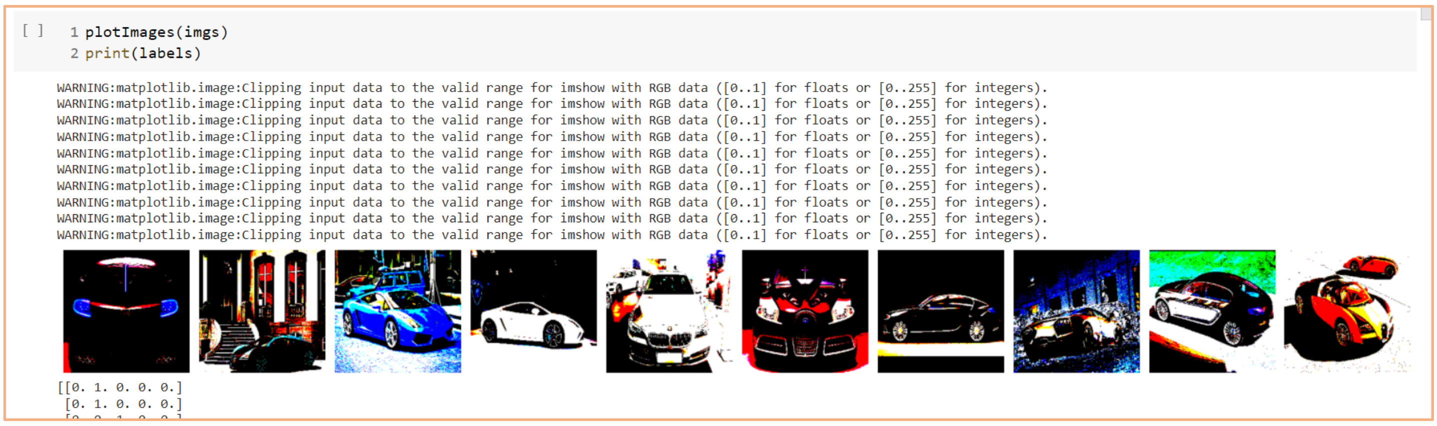


```
imgs, labels = next(train_batches)
print(type(imgs))
test_batches.class_indices # BMW @ index 0 , Bugatti @ index 1, Lam@ index 2, Mclaren@ index 3, Volkswagen@ index 4
```

```
<class 'numpy.ndarray'>
{'BMW': 0, 'Bugatti': 1, 'Lamborghini': 2, 'McLaren': 3, 'Volkswagen': 4}
```



```
[ ]
# This function will plot images in the form of a grid with 1 row and 10 columns
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```



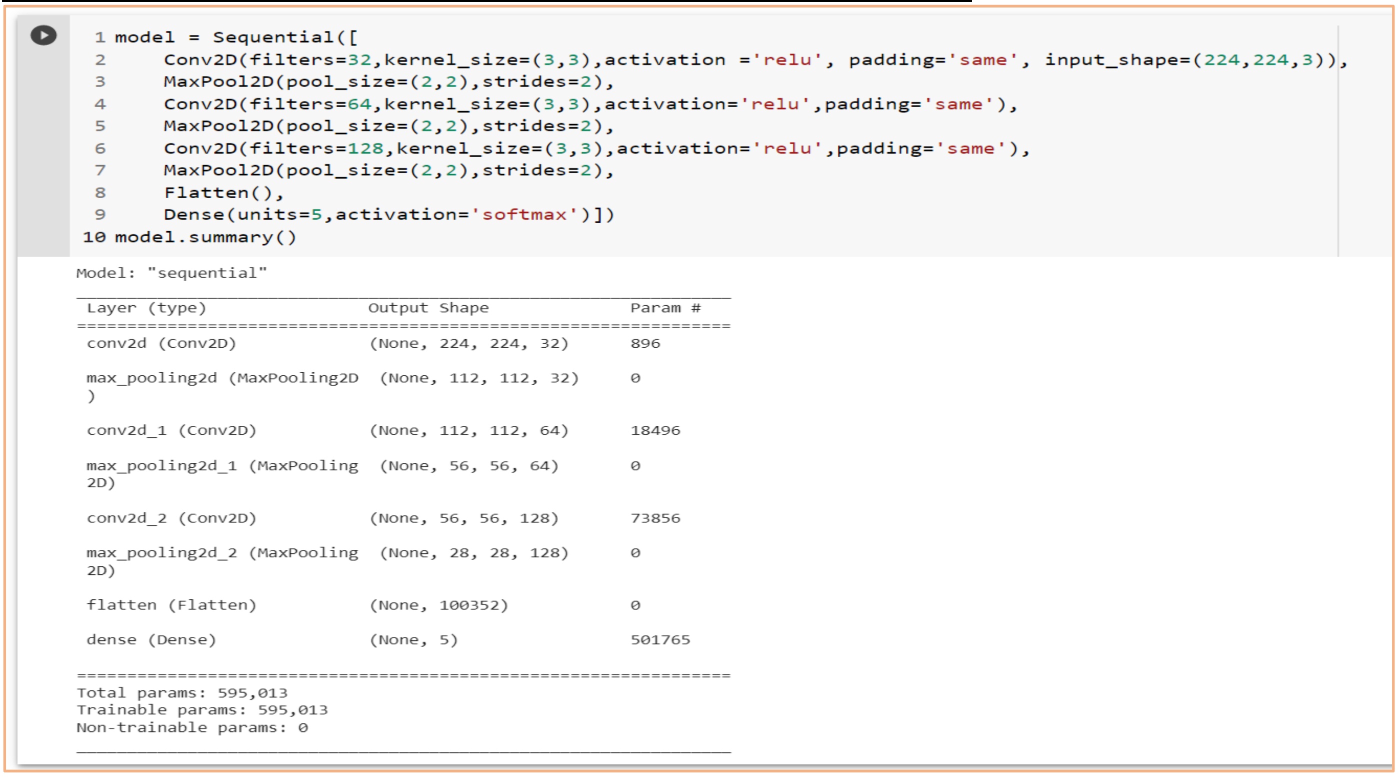
```
[ ]
plotImages(imgs)
print(labels)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
[[0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
```

b) Build a convolutional neural network either from scratch or existing models

Screenshots of new convolutional neural network from scratch:



```
model = Sequential([
    Conv2D(filters=32,kernel_size=(3,3),activation ='relu', padding='same', input_shape=(224,224,3)),
    MaxPool2D(pool_size=(2,2),strides=2),
    Conv2D(filters=64,kernel_size=(3,3),activation='relu',padding='same'),
    MaxPool2D(pool_size=(2,2),strides=2),
    Conv2D(filters=128,kernel_size=(3,3),activation='relu',padding='same'),
    MaxPool2D(pool_size=(2,2),strides=2),
    Flatten(),
    Dense(units=5,activation='softmax')])
model.summary()
```

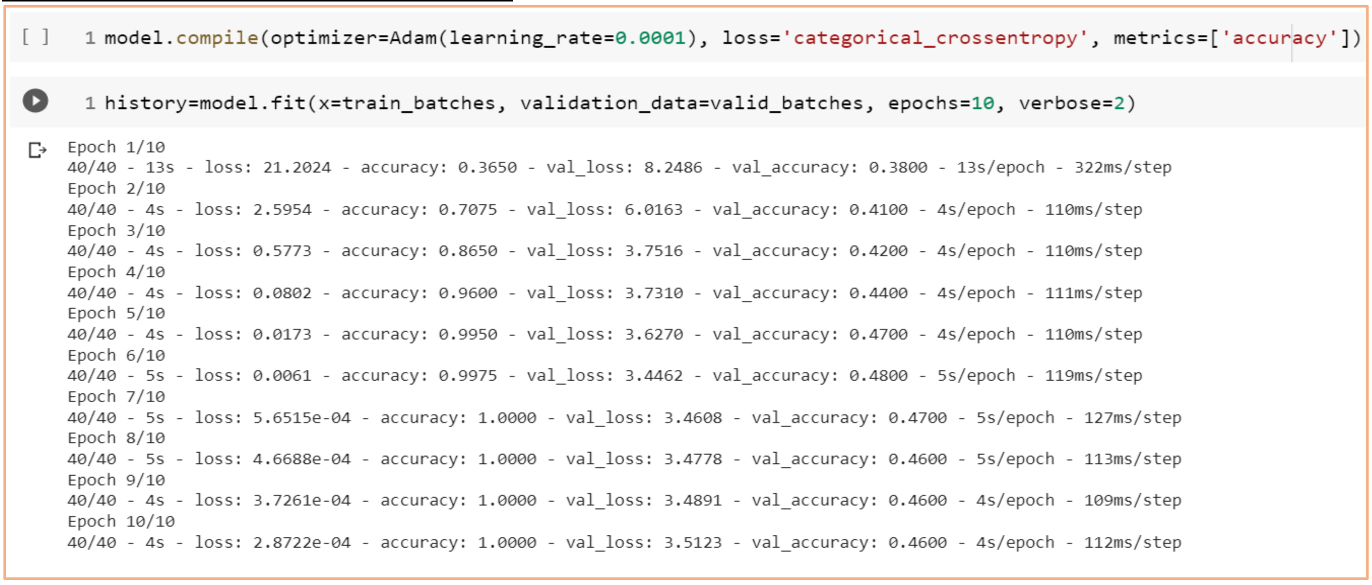
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 56, 56, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 100352)            0         
                                                                 
 dense (Dense)               (None, 5)                 501765    
                                                                 
=================================================================
Total params: 595,013
Trainable params: 595,013
Non-trainable params: 0
_________________________________________________________________
```

3 pairs of Conv2D with each having 32,64 and 128 output filters using ReLU as activation function & MaxPool2D had been layered on top of each other followed by a flatten and dense layer of 5 units using “softmax” as activation function. ReLU function returns all negative values as 0 and maximum positive number is taken in for consideration. With its linearity feature for positive numbers, it allows scaling of the CNN network without having exponential growth in the computation power needed. Softmax function is most suitable for multi-classification model as it returns the probabilities of each class & target class that has the highest probability.

Name: Lim Junwei Darien Matriculation No.: U1921811F

c) <u>Train the neural network, evaluate the model prediction, and obtain the evaluation metrics, including prediction accuracy and confusion matrix</u>

<u>Screenshots of training of model:</u>



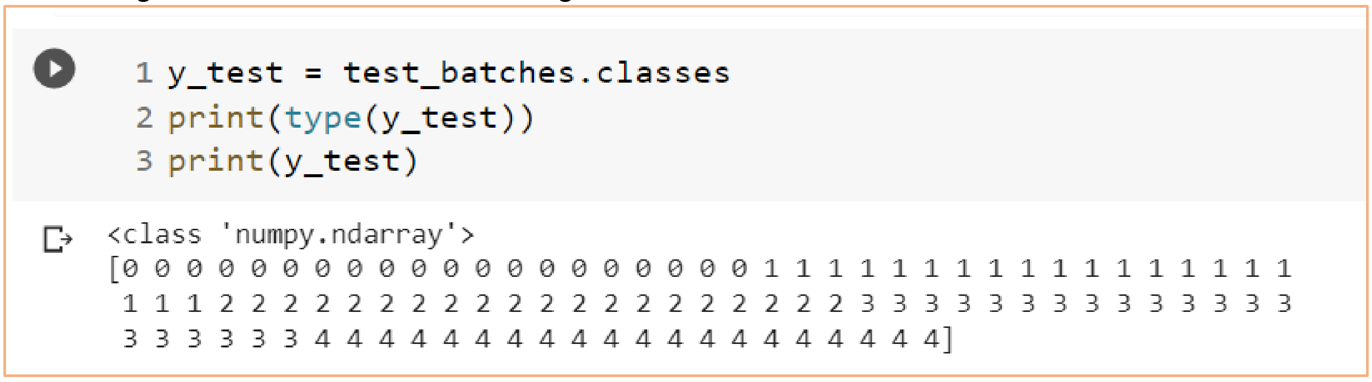
```
[ ] model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
history=model.fit(x=train_batches, validation_data=valid_batches, epochs=10, verbose=2)
```

```
Epoch 1/10
40/40 - 13s - loss: 21.2024 - accuracy: 0.3650 - val_loss: 8.2486 - val_accuracy: 0.3800 - 13s/epoch - 322ms/step
Epoch 2/10
40/40 - 4s - loss: 2.5954 - accuracy: 0.7075 - val_loss: 6.0163 - val_accuracy: 0.4100 - 4s/epoch - 110ms/step
Epoch 3/10
40/40 - 4s - loss: 0.5773 - accuracy: 0.8650 - val_loss: 3.7516 - val_accuracy: 0.4200 - 4s/epoch - 110ms/step
Epoch 4/10
40/40 - 4s - loss: 0.0802 - accuracy: 0.9600 - val_loss: 3.7310 - val_accuracy: 0.4400 - 4s/epoch - 111ms/step
Epoch 5/10
40/40 - 4s - loss: 0.0173 - accuracy: 0.9950 - val_loss: 3.6270 - val_accuracy: 0.4700 - 4s/epoch - 110ms/step
Epoch 6/10
40/40 - 5s - loss: 0.0061 - accuracy: 0.9975 - val_loss: 3.4462 - val_accuracy: 0.4800 - 5s/epoch - 119ms/step
Epoch 7/10
40/40 - 5s - loss: 5.6515e-04 - accuracy: 1.0000 - val_loss: 3.4608 - val_accuracy: 0.4700 - 5s/epoch - 127ms/step
Epoch 8/10
40/40 - 5s - loss: 4.6688e-04 - accuracy: 1.0000 - val_loss: 3.4778 - val_accuracy: 0.4600 - 5s/epoch - 113ms/step
Epoch 9/10
40/40 - 4s - loss: 3.7261e-04 - accuracy: 1.0000 - val_loss: 3.4891 - val_accuracy: 0.4600 - 4s/epoch - 109ms/step
Epoch 10/10
40/40 - 4s - loss: 2.8722e-04 - accuracy: 1.0000 - val_loss: 3.5123 - val_accuracy: 0.4600 - 4s/epoch - 112ms/step
```

<u>Screenshots of evaluation of the model prediction:</u>

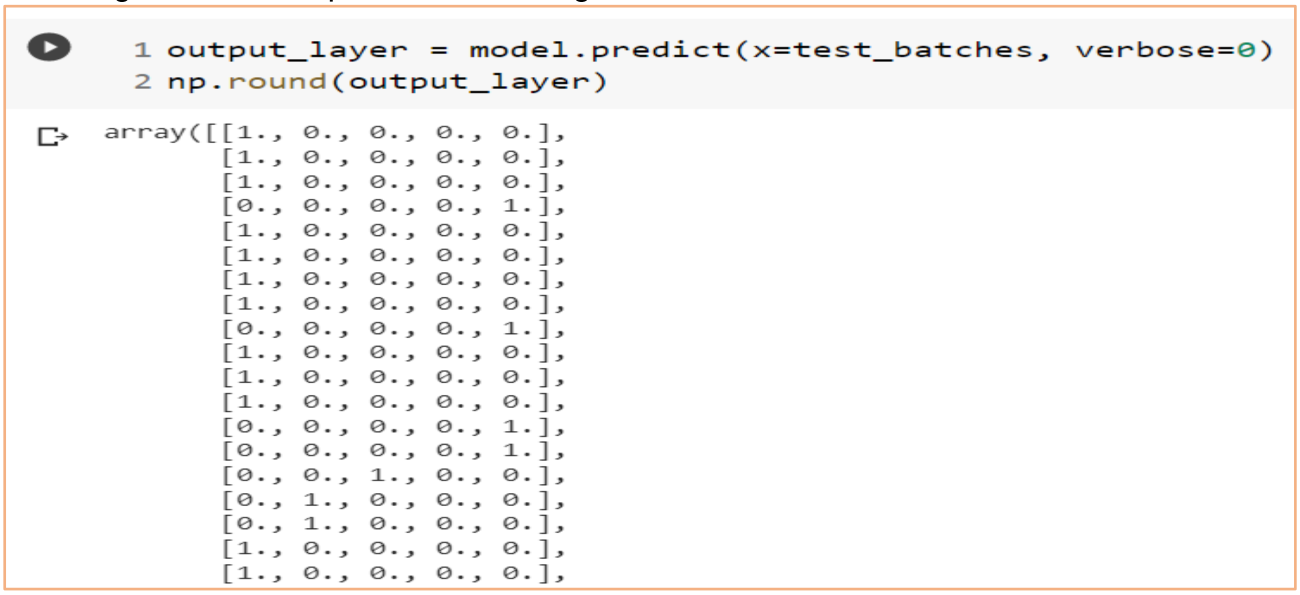
- Getting the true label of the test images



```
y_test = test_batches.classes
print(type(y_test))
print(y_test)
```

```
<class 'numpy.ndarray'>
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4]
```

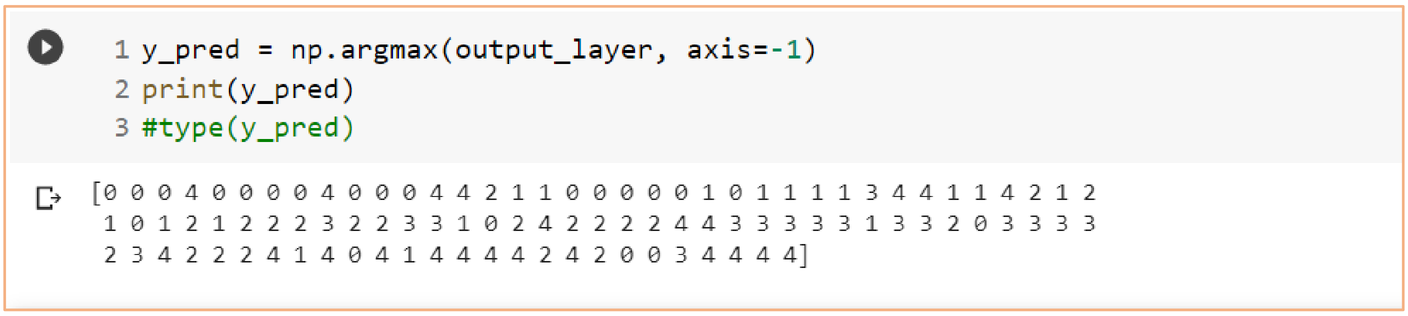
- Using our model to predict the 5 categories



```
output_layer = model.predict(x=test_batches, verbose=0)
np.round(output_layer)
```

```
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
```

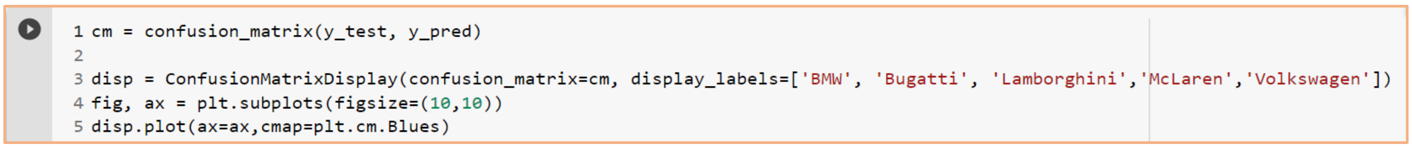
Name: Lim Junwei Darien Matriculation No.: U1921811F



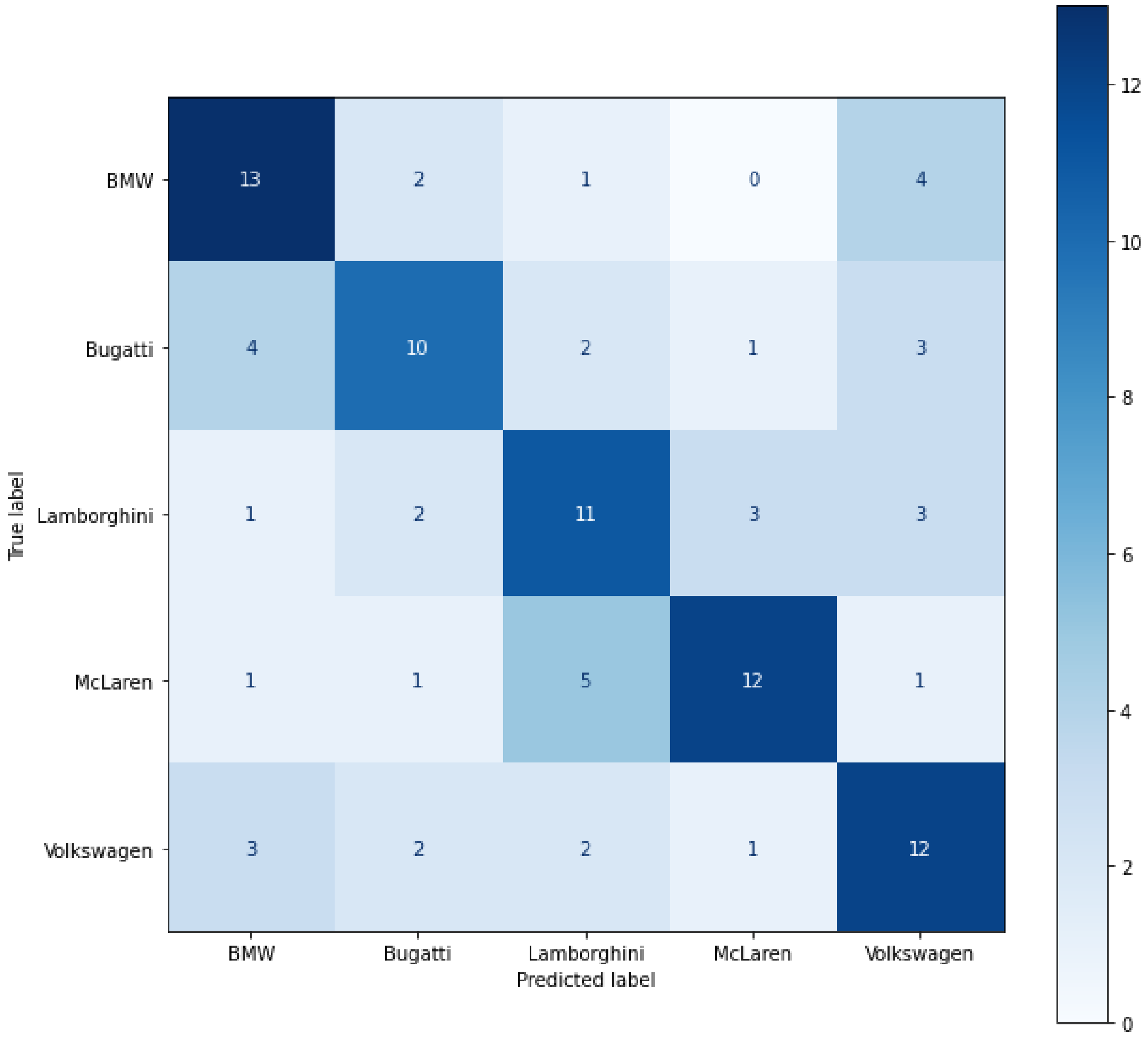
```
y_pred = np.argmax(output_layer, axis=-1)
print(y_pred)
#type(y_pred)
```

```
[0 0 0 4 0 0 0 0 4 0 0 0 4 4 2 1 1 0 0 0 0 0 1 0 1 1 1 1 3 4 4 1 1 4 2 1 2
 1 0 1 2 1 2 2 2 3 2 2 3 3 1 0 2 4 2 2 2 2 4 4 3 3 3 3 3 1 3 3 2 0 3 3 3 3
 2 3 4 2 2 2 4 1 4 0 4 1 4 4 4 4 2 4 2 0 0 3 4 4 4 4]
```

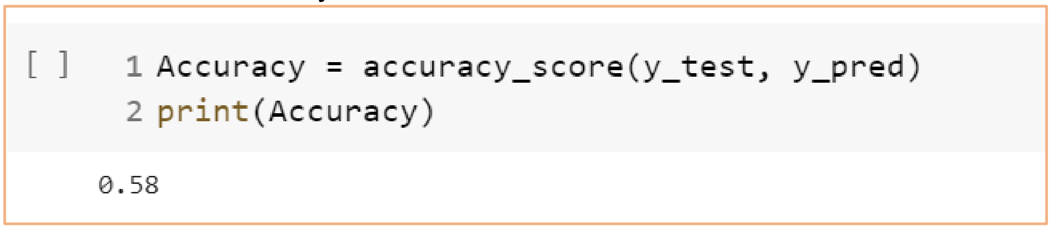
- Plotting of Confusion Matrix



```
cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'])
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax,cmap=plt.cm.Blues)
```



- Prediction Accuracy of 58%



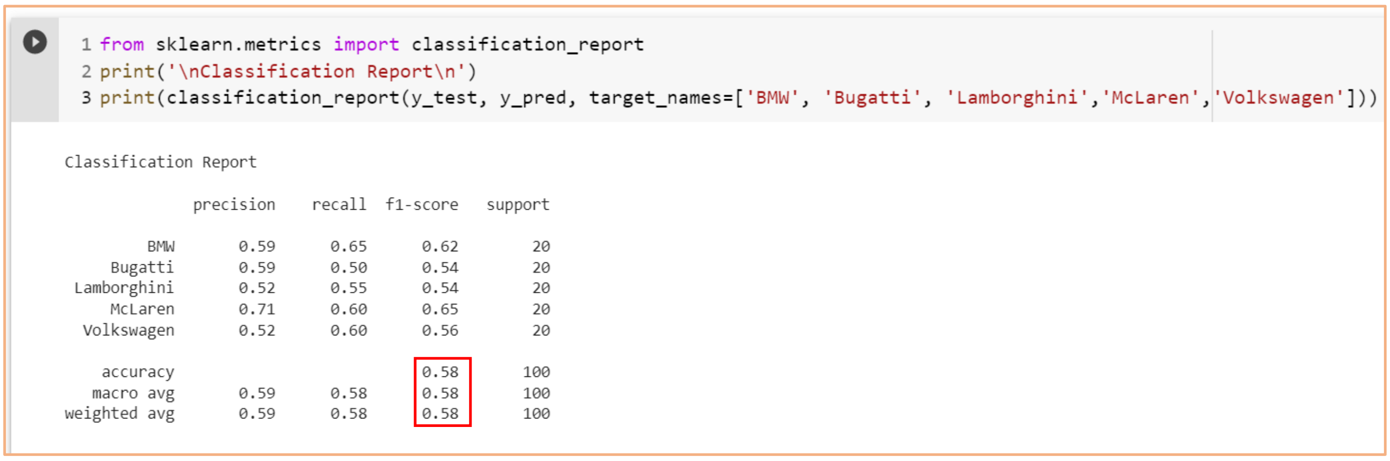
```
[ ]  Accuracy = accuracy_score(y_test, y_pred)
     print(Accuracy)

0.58
```

d) Based on error analysis and online materials, suggest a proper direction that can lead to better performance of your model in your presentation document

1) **Error Analysis**

- Screenshot of getting F_1-Score

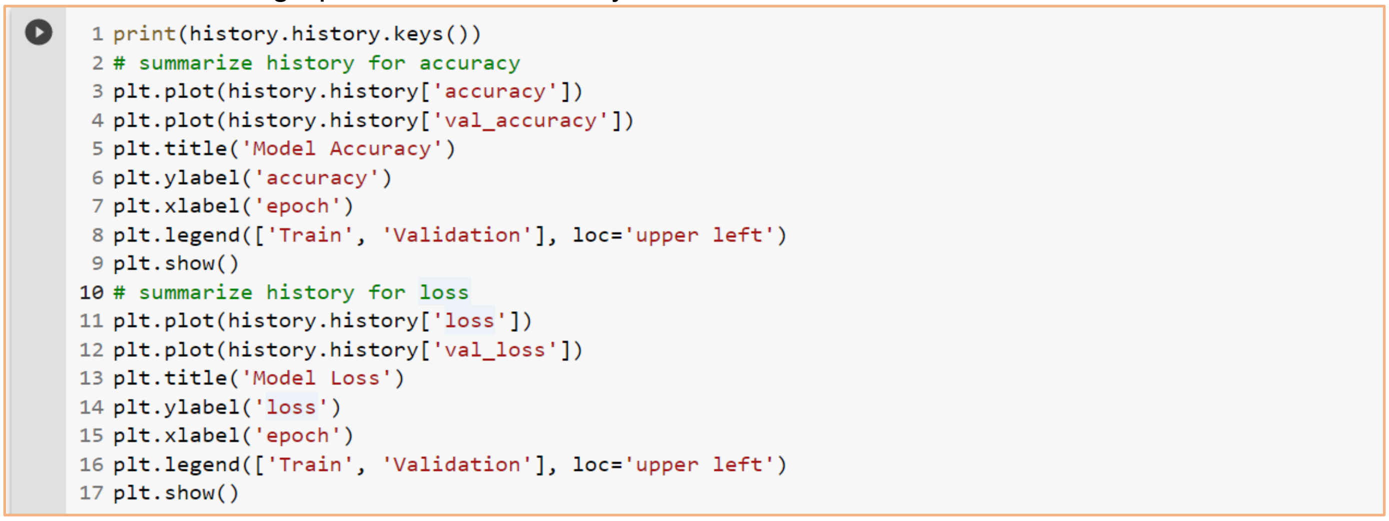
Code is from [1]:



```
from sklearn.metrics import classification_report
print('\nClassification Report\n')
print(classification_report(y_test, y_pred, target_names=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen']))

Classification Report

              precision    recall  f1-score   support

         BMW       0.59      0.65      0.62        20
     Bugatti       0.59      0.50      0.54        20
 Lamborghini       0.52      0.55      0.54        20
     McLaren       0.71      0.60      0.65        20
  Volkswagen       0.52      0.60      0.56        20

    accuracy                           0.58       100
   macro avg       0.59      0.58      0.58       100
weighted avg       0.59      0.58      0.58       100
```

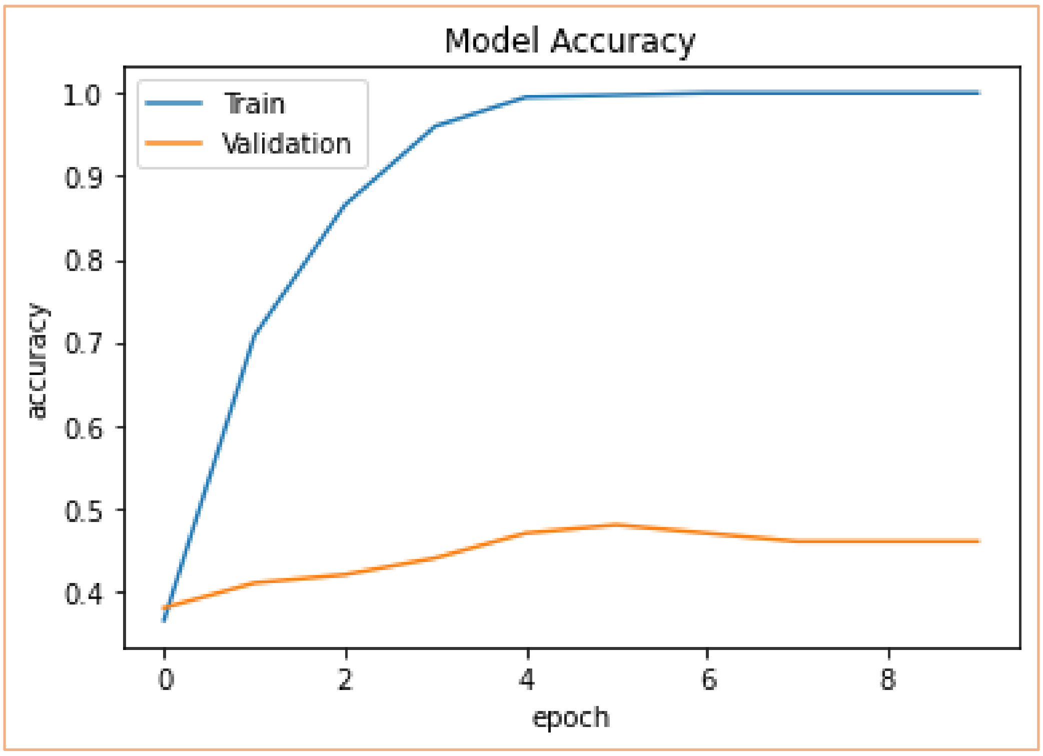
Macro and Weighted average F_1 score is 0.58 as all the 5 classes have 20 images each with an accuracy of 58% of the test data set that was stated in part c. Among the classes, McLaren seems to have the highest F_1-score which shows that the model is better at identifying McLaren thus indicating certain level of biasness. Since the average F_1 score is not near 1 and low accuracy overall, the model is not being generalised well enough.

- Screenshots of graph of Model Accuracy and Model Loss

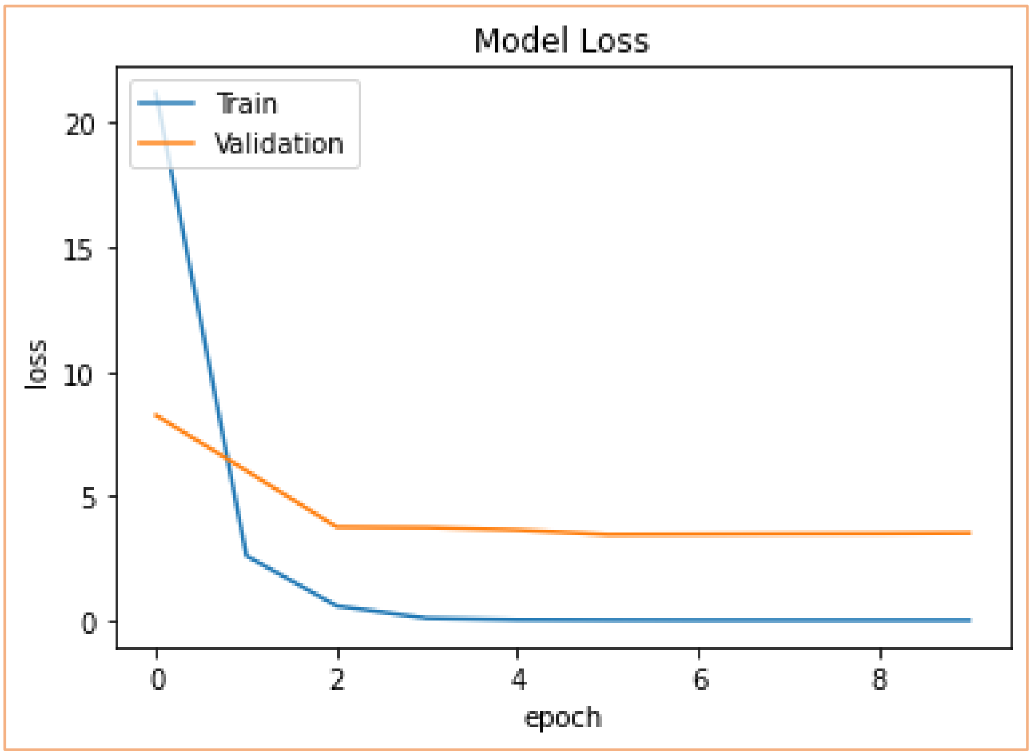


```
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

- Graph of Model Accuracy



- Graph of Model Loss



Based on the accuracy graph above, the accuracy of the model has hit a plateau for both training and validation data set. In addition, training set accuracy is also higher than validations set, this would mean the model is overfitting. As for the loss graph, there is no significant reduction after reaching 1 epoch on the validation data. Thus, further increasing of the epoch no. is not recommended for these settings of the model.

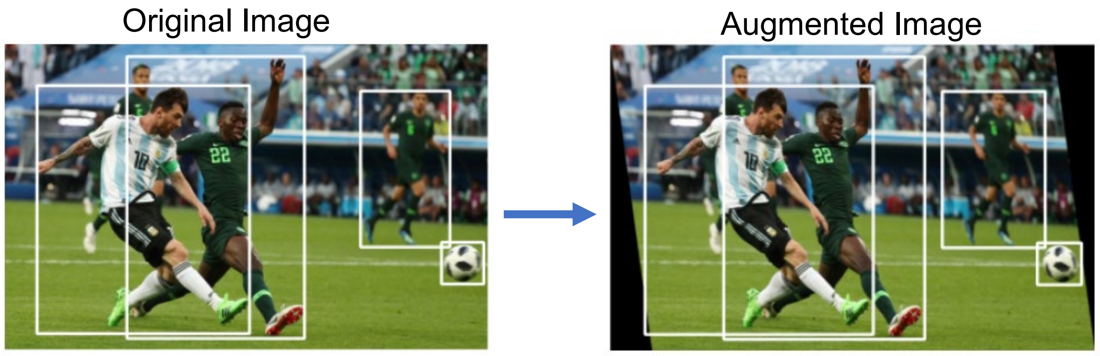
Name: Lim Junwei Darien Matriculation No.: U1921811F

2) Online materials

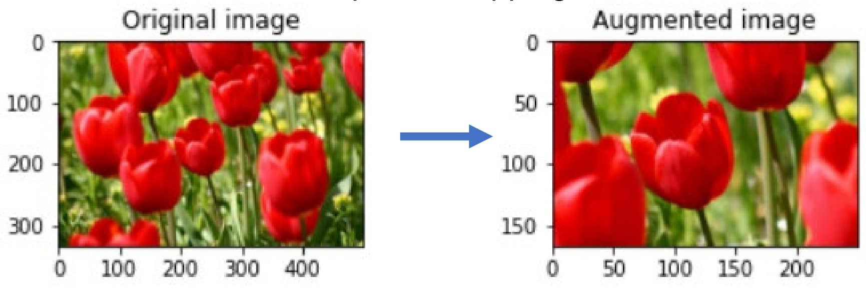
The model would require more generalization as there is an overfitting situation currently. Regularization can prevent overfitting thus improving generalization. One of the ways is through data augmentation with the given fixed set of data, thus the model will not see the same picture twice [2],[3],[4]. Data augmentation is a preprocessing technique where there is cropping, flipping, zooming, shearing, translation and etc of the original images [3].

Images taken from [5],[6],[7]

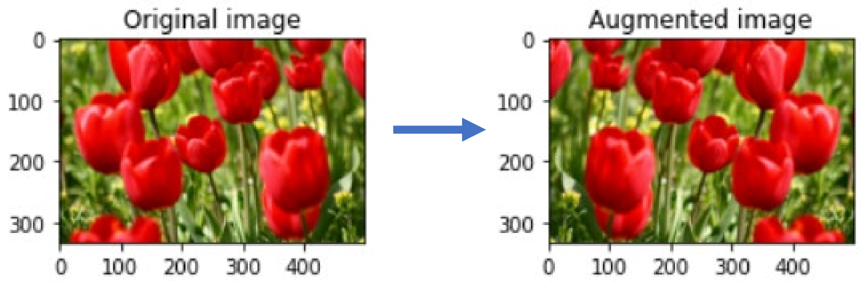
Example of Shearing:



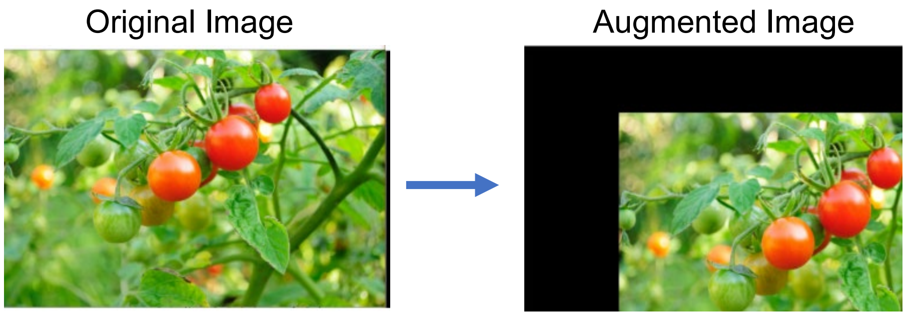
Example of Cropping:



Example of Flipping:



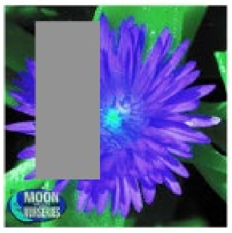
Example of Translation:



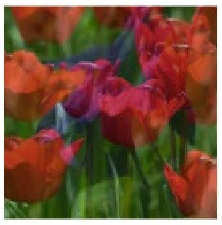
There are also other data augmentation methods, such as CutMix and MixUp that can generate high-quality inter-class images. CutMix randomly crops portions of one image and overlay onto another image whereas MixUp interpolates pixel values values between 2 images, it has a result of a faded layer onto another image [8].

Images taken from [9]

Example of CutMix:



Example of MixUp:



However, either CutMix or MixUp is usually implemented in most machine learning models not at the same time [8].

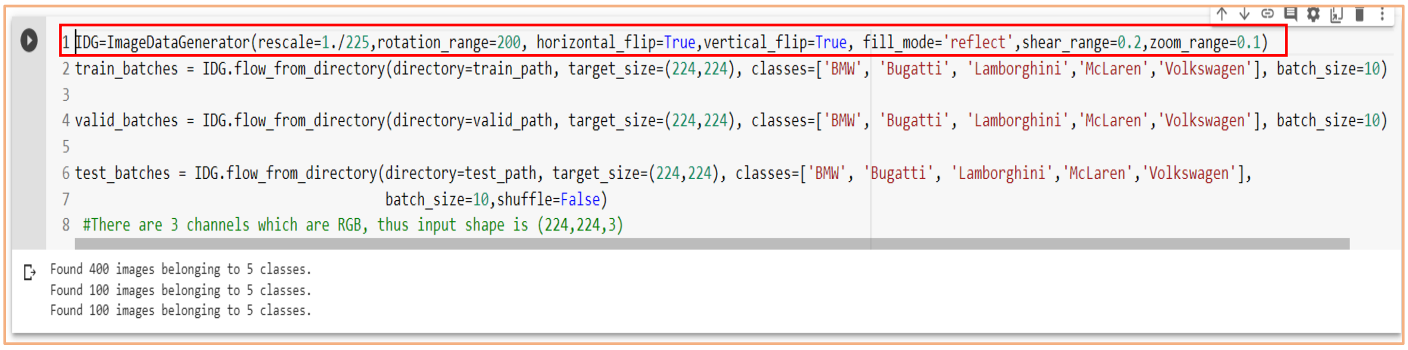
Name: Lim Junwei Darien Matriculation No.: U1921811F

e) Implement the new direction suggested in d) and demonstrate better evaluation metrics in your prediction outcome

1) **Implementing several changes based on data augmentation to improve the metrics which are:**

1.1 Changing ImageDataGenerator()

The implementation below was reference from [4].



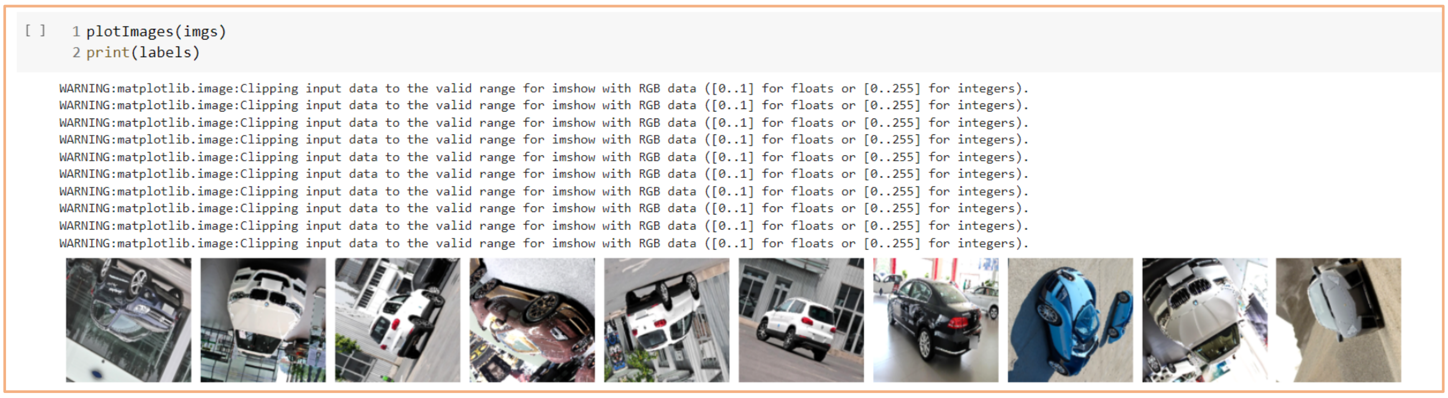
```
IDG=ImageDataGenerator(rescale=1./225,rotation_range=200, horizontal_flip=True,vertical_flip=True, fill_mode='reflect',shear_range=0.2,zoom_range=0.1)
train_batches = IDG.flow_from_directory(directory=train_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'], batch_size=10)

valid_batches = IDG.flow_from_directory(directory=valid_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'], batch_size=10)

test_batches = IDG.flow_from_directory(directory=test_path, target_size=(224,224), classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'],
                                       batch_size=10,shuffle=False)
 #There are 3 channels which are RGB, thus input shape is (224,224,3)
```

```
Found 400 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
```

ImageDataGenerator() have included for all training, validation and test batches :

- Rescaling of 1/255 factor, this is to reduce the RGB coefficients between 0 and 1 for easy processing by the model.
- Random rotation of 200°
- Horizontal & vertical flipping
- Random shearing of 0.2 in intensity and random zooming in of of the original
- Random zooming of 0.1 ([1-zoom_range, 1+zoom_range], where zoom_range is 0.1)

These are the results:



```
plotImages(imgs)
print(labels)
```

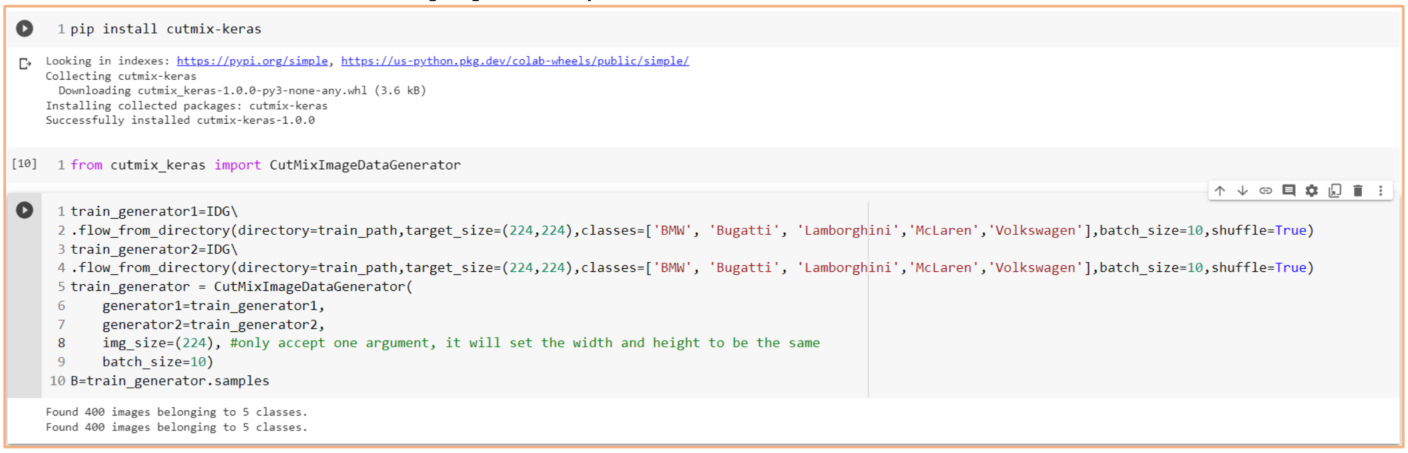
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

It shows that the images have successfully been augmented.

Name: Lim Junwei Darien Matriculation No.: U1921811F

1.2 Adding CutMix to our training set after ImageDataGenerator()

The code was reference from [10] and implemented which is shown below:



```
pip install cutmix-keras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting cutmix-keras
  Downloading cutmix_keras-1.0.0-py3-none-any.whl (3.6 kB)
Installing collected packages: cutmix-keras
Successfully installed cutmix-keras-1.0.0
```

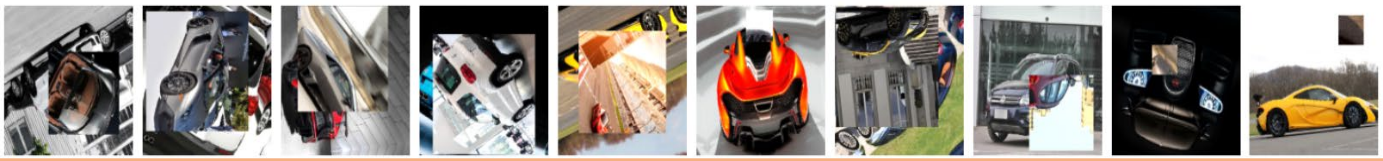
```
[10] from cutmix_keras import CutMixImageDataGenerator
```

```
train_generator1=IDG\
.flow_from_directory(directory=train_path,target_size=(224,224),classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'],batch_size=10,shuffle=True)
train_generator2=IDG\
.flow_from_directory(directory=train_path,target_size=(224,224),classes=['BMW', 'Bugatti', 'Lamborghini','McLaren','Volkswagen'],batch_size=10,shuffle=True)
train_generator = CutMixImageDataGenerator(
    generator1=train_generator1,
    generator2=train_generator2,
    img_size=(224), #only accept one argument, it will set the width and height to be the same
    batch_size=10)
B=train_generator.samples
```

```
Found 400 images belonging to 5 classes.
Found 400 images belonging to 5 classes.
```

These are the results:

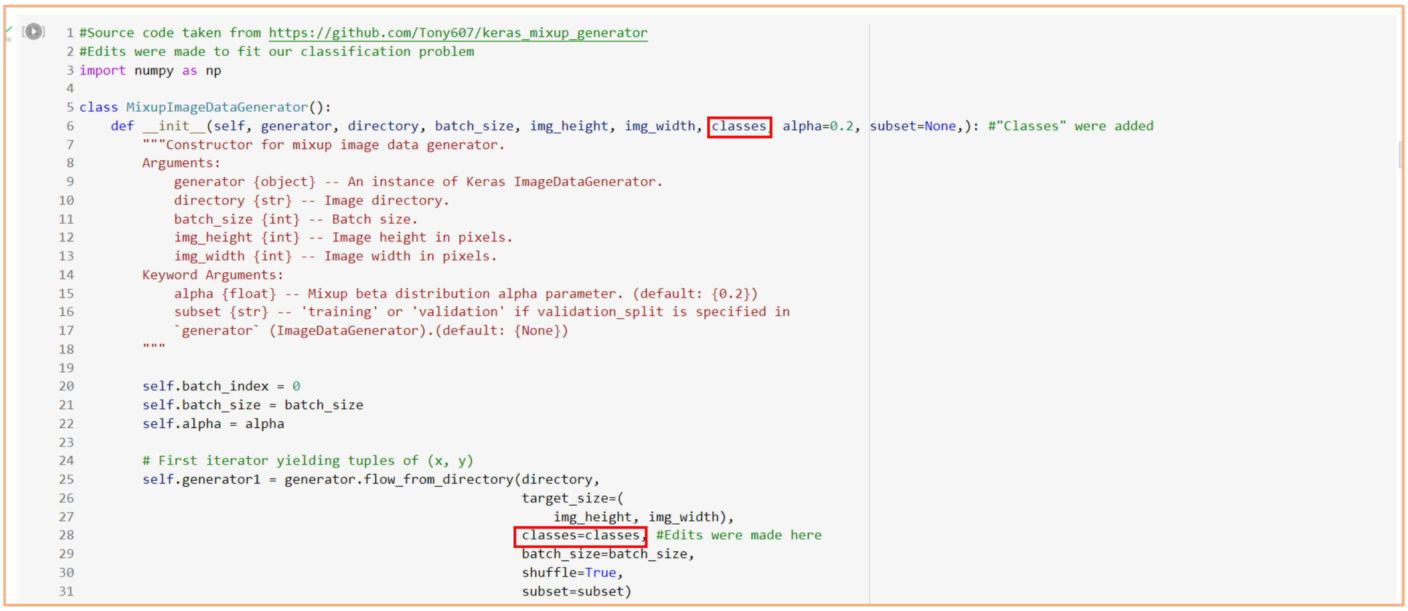
```
plotImages(img)
print(label)
```



It shows that the images have successfully been cut & mix

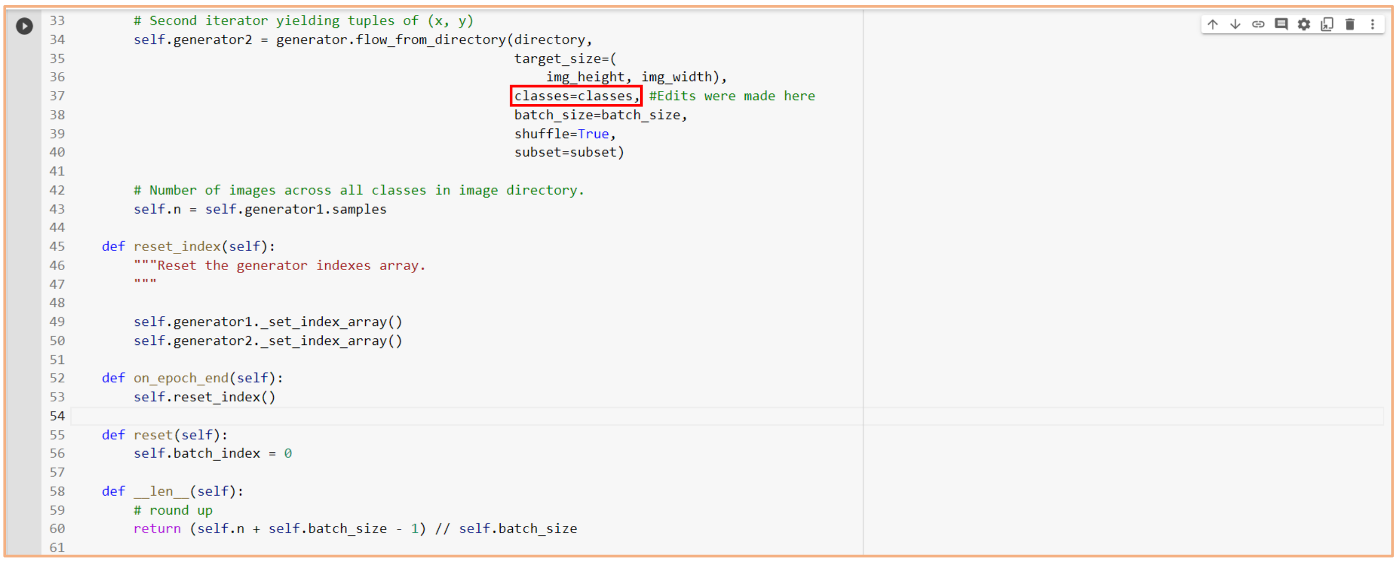
1.3 Adding MixUp to our training set after ImageDataGenerator() as an another alternative method to compare with CutMix performance

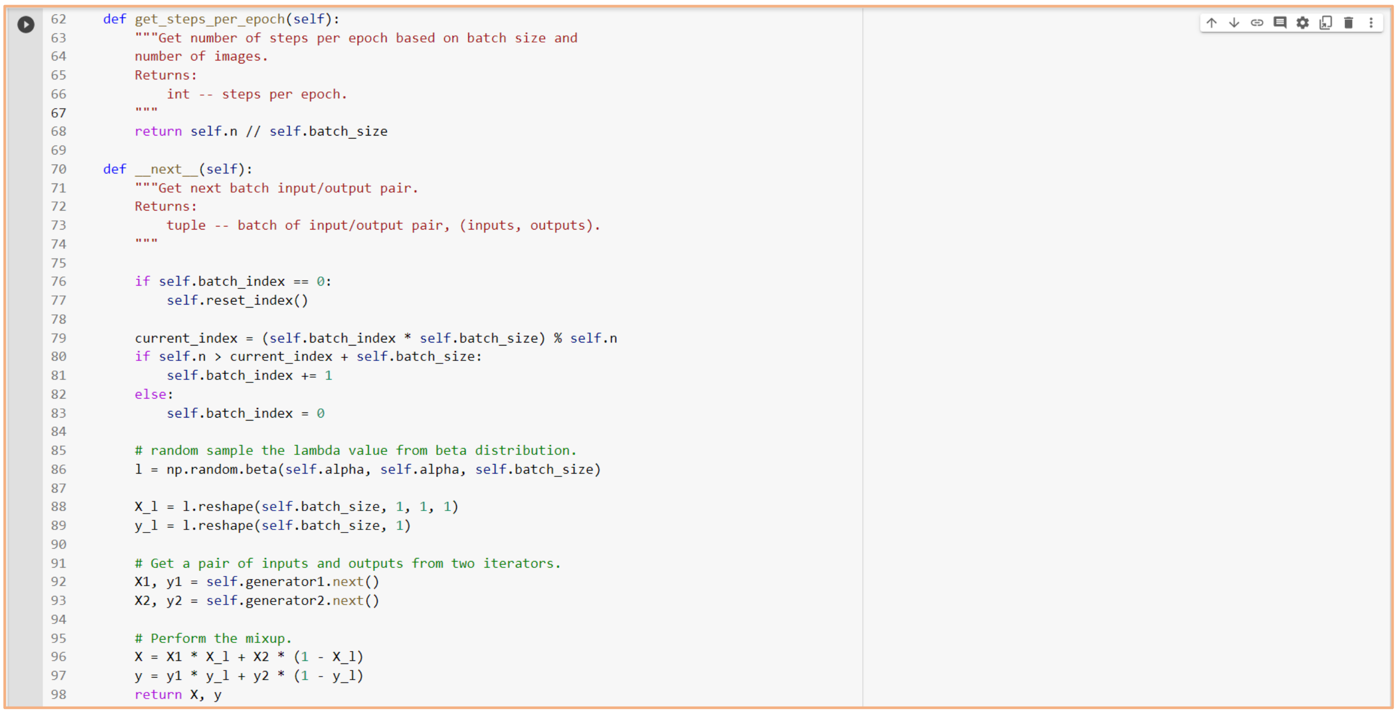
The source code was reference from [11] and changes (red boxes) were made to fit our classification problem before being implemented which is shown below:

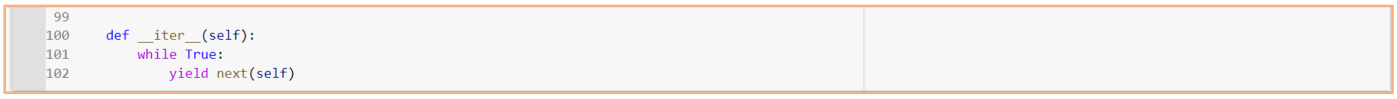


```
#Source code taken from https://github.com/Tony607/keras_mixup_generator
#Edits were made to fit our classification problem
import numpy as np

class MixupImageDataGenerator():
    def __init__(self, generator, directory, batch_size, img_height, img_width, classes, alpha=0.2, subset=None,): #"Classes" were added
        """Constructor for mixup image data generator.
        Arguments:
            generator {object} -- An instance of Keras ImageDataGenerator.
            directory {str} -- Image directory.
            batch_size {int} -- Batch size.
            img_height {int} -- Image height in pixels.
            img_width {int} -- Image width in pixels.
        Keyword Arguments:
            alpha {float} -- Mixup beta distribution alpha parameter. (default: {0.2})
            subset {str} -- 'training' or 'validation' if validation_split is specified in
            `generator` (ImageDataGenerator).(default: {None})
        """

        self.batch_index = 0
        self.batch_size = batch_size
        self.alpha = alpha

        # First iterator yielding tuples of (x, y)
        self.generator1 = generator.flow_from_directory(directory,
                                                        target_size=(
                                                            img_height, img_width),
                                                        classes=classes, #Edits were made here
                                                        batch_size=batch_size,
                                                        shuffle=True,
                                                        subset=subset)
```

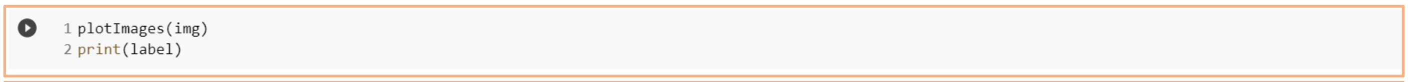
Name: Lim Junwei Darien Matriculation No.: U1921811F

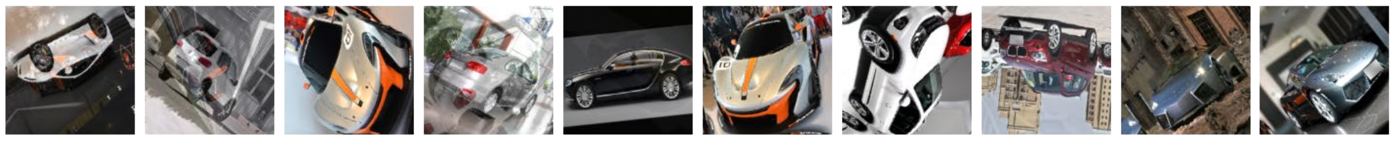

```python
        # Second iterator yielding tuples of (x, y)
        self.generator2 = generator.flow_from_directory(directory,
                                                        target_size=(
                                                            img_height, img_width),
                                                        classes=classes, #Edits were made here
                                                        batch_size=batch_size,
                                                        shuffle=True,
                                                        subset=subset)

        # Number of images across all classes in image directory.
        self.n = self.generator1.samples

    def reset_index(self):
        """Reset the generator indexes array.
        """

        self.generator1._set_index_array()
        self.generator2._set_index_array()

    def on_epoch_end(self):
        self.reset_index()

    def reset(self):
        self.batch_index = 0

    def __len__(self):
        # round up
        return (self.n + self.batch_size - 1) // self.batch_size

```


```python
    def get_steps_per_epoch(self):
        """Get number of steps per epoch based on batch size and
        number of images.
        Returns:
            int -- steps per epoch.
        """
        return self.n // self.batch_size

    def __next__(self):
        """Get next batch input/output pair.
        Returns:
            tuple -- batch of input/output pair, (inputs, outputs).
        """

        if self.batch_index == 0:
            self.reset_index()

        current_index = (self.batch_index * self.batch_size) % self.n
        if self.n > current_index + self.batch_size:
            self.batch_index += 1
        else:
            self.batch_index = 0

        # random sample the lambda value from beta distribution.
        l = np.random.beta(self.alpha, self.alpha, self.batch_size)

        X_l = l.reshape(self.batch_size, 1, 1, 1)
        y_l = l.reshape(self.batch_size, 1)

        # Get a pair of inputs and outputs from two iterators.
        X1, y1 = self.generator1.next()
        X2, y2 = self.generator2.next()

        # Perform the mixup.
        X = X1 * X_l + X2 * (1 - X_l)
        y = y1 * y_l + y2 * (1 - y_l)
        return X, y
```


```python

    def __iter__(self):
        while True:
            yield next(self)
```

These are the results:

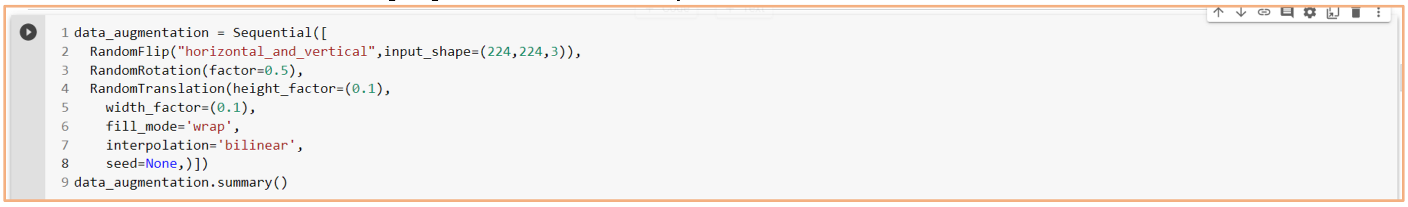

```python
plotImages(img)
print(label)
```



It shows that the images have successfully been mixup with a faded overlay of original images.

1.4 Adding data augmentation as a layer into our model

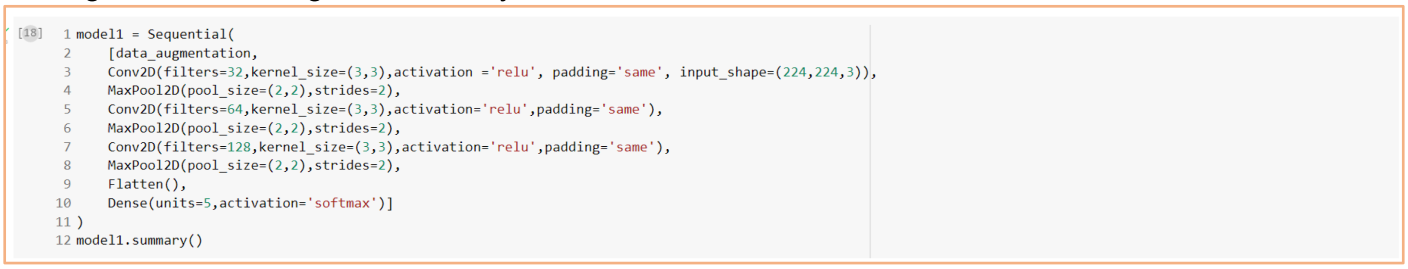
Reference was made from [12] for the below implementation:



```
data_augmentation = Sequential([
  RandomFlip("horizontal_and_vertical",input_shape=(224,224,3)),
  RandomRotation(factor=0.5),
  RandomTranslation(height_factor=(0.1),
    width_factor=(0.1),
    fill_mode='wrap',
    interpolation='bilinear',
    seed=None,)])
data_augmentation.summary()
```

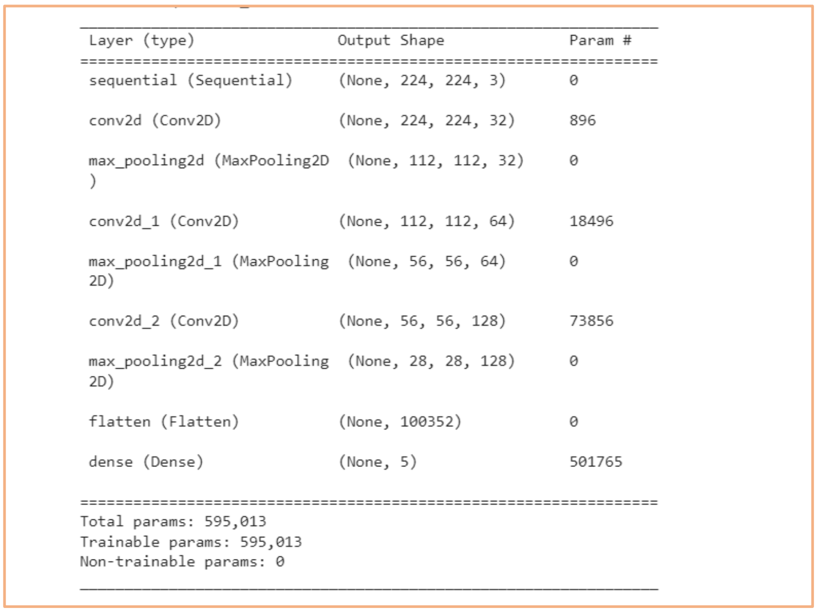
3 different augmentations have been added as preprocessing layers, which allow further data augmentation whenever the training dataset passes through the model for every iteration.

Adding of the data augmentation layer into our current model:



```
[18]
model1 = Sequential(
    [data_augmentation,
    Conv2D(filters=32,kernel_size=(3,3),activation ='relu', padding='same', input_shape=(224,224,3)),
    MaxPool2D(pool_size=(2,2),strides=2),
    Conv2D(filters=64,kernel_size=(3,3),activation='relu',padding='same'),
    MaxPool2D(pool_size=(2,2),strides=2),
    Conv2D(filters=128,kernel_size=(3,3),activation='relu',padding='same'),
    MaxPool2D(pool_size=(2,2),strides=2),
    Flatten(),
    Dense(units=5,activation='softmax')]
)
model1.summary()
```

This is the result:



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
 2D)

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 5)                 501765

=================================================================
Total params: 595,013
Trainable params: 595,013
Non-trainable params: 0
_________________________________________________________________
```

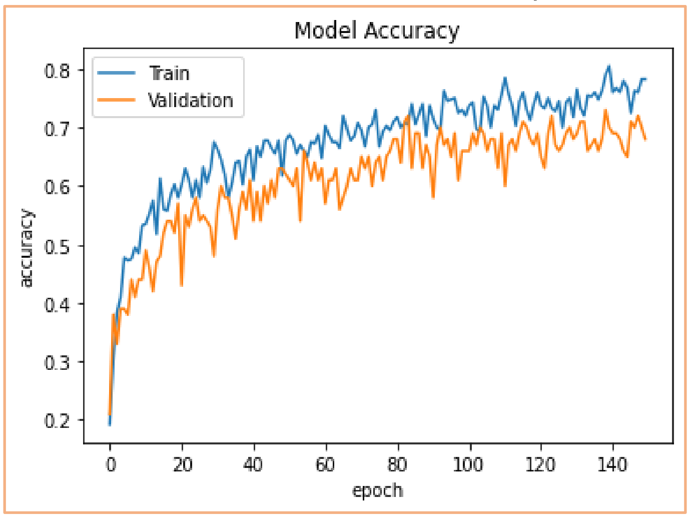
2) **Demonstration of better evaluation metrics in prediction outcome**

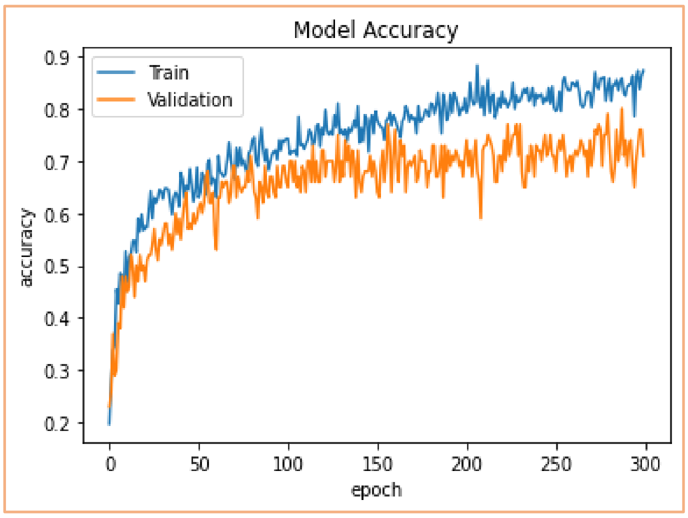
Mixup & Cutmix are compared to choose the better method that gives us higher prediction accuracy.

2.1 Mixup Implementation

2.1.1 Model Accuracy

The model was trained in 2 different epochs

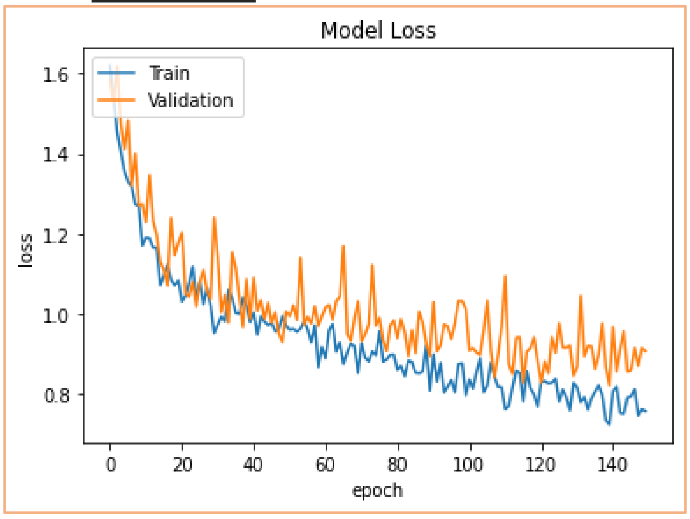


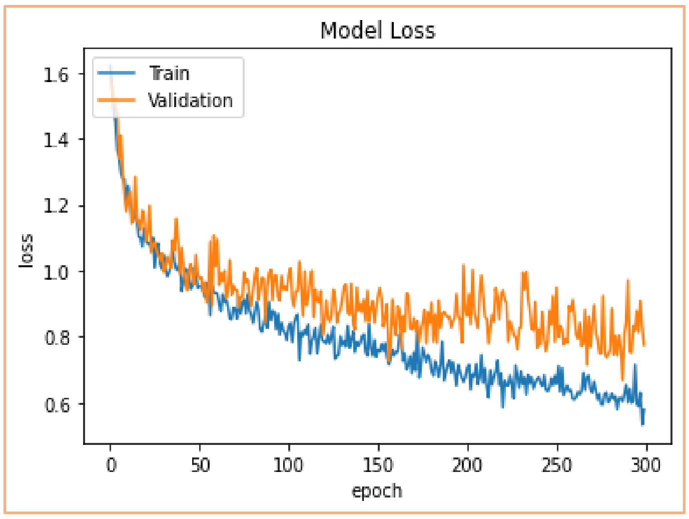


Graph 1 at 150 Epoch Graph 2 at 300 Epoch

The accuracy graphs show increasing trends for both training and validation set with minimal difference between them for graph 1 which suggested that the model is a good fit, thus this motivates the continual increasing of epoch no. to get higher accuracy. However, at graph 2 with 300 epoch the accuracy between the training and validation set starts to diverge at about 200 epochs with training set exceeding the validation set this indicates the starting of an overfitting model. Therefore, epochs no. below 200 but above 150 can provide a good fit for our future prediction.

2.1.2 Model Loss

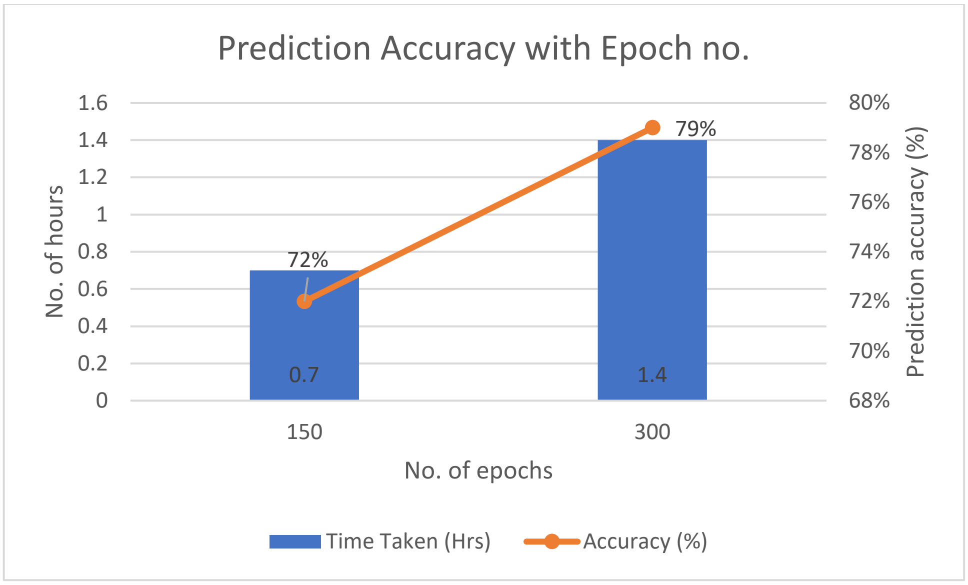




Graph 1 at 150 Epoch Graph 2 at 300 Epoch

The graph 1 show that there is a decreasing trend happening for both training and validation set losses, which prompted the increase of epoch no. as well which coincides with the accuracy graphs 1. Graph 2 with 300 epochs shows that there is a divergence at about 200 epochs with the validation set loss starts to increase while training set loss continue to decrease, which are signs of overfitting.

2.1.3 Prediction Accuracy

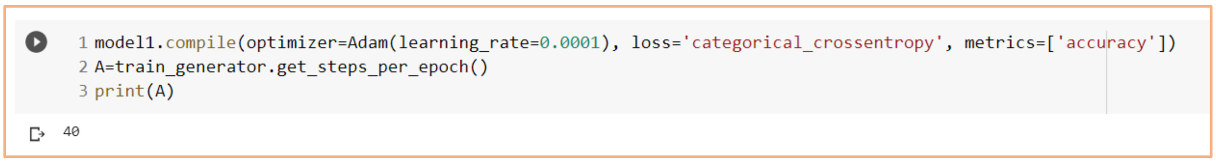


Graph of prediction accuracy with epoch no & hours for Mixup.

The graph above overall shows that there is an increasing trend of the prediction accuracy when there is an increase of epochs no. which indicates that increasing epochs no. increases the model prediction accuracy. The incremental change of the prediction accuracy is significant of 7% between epochs 150 & 300 with twice the time spent training at 150 epochs. However, the maximum of epoch would be 300 as further increase of the epoch would result in overfitting of the model and decrease the prediction accuracy, therefore the highest accuracy for this model is 79% at 300 epochs.

2.2 Cutmix Implementation

An additional step is required to extract the steps required per epoch from Cutmix generator shown in part e,1.2 as the function is not able to return the steps required per epoch for the fitting of the model later on.

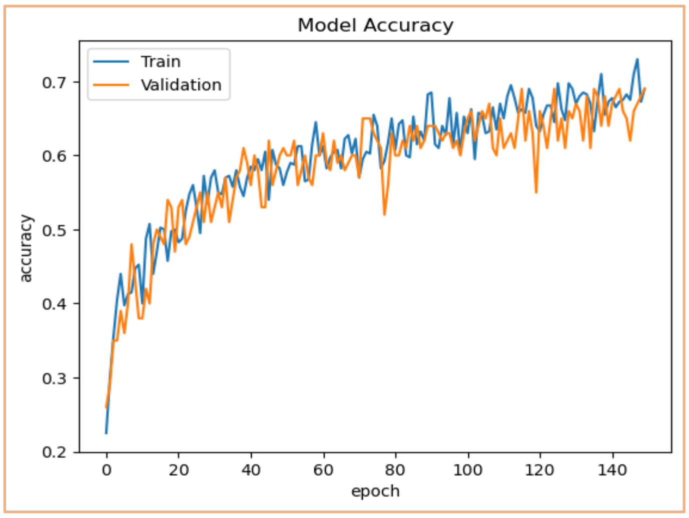


```
model1.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
A=train_generator.get_steps_per_epoch()
print(A)
```

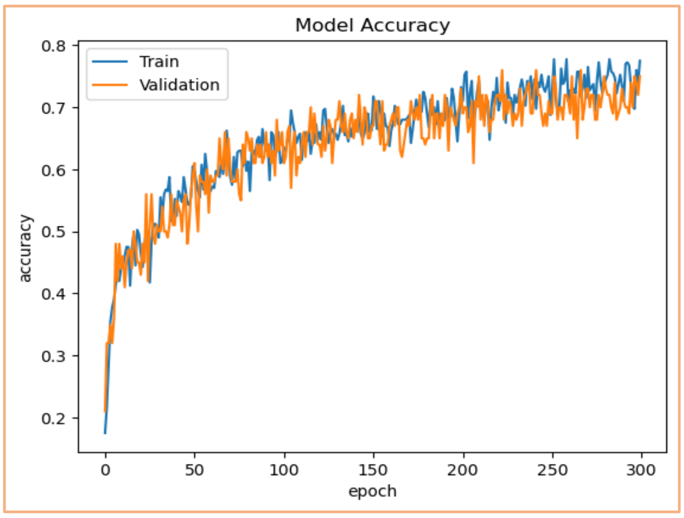
```
40
```

The model was trained in 4 different epochs, 150,300,500,800:

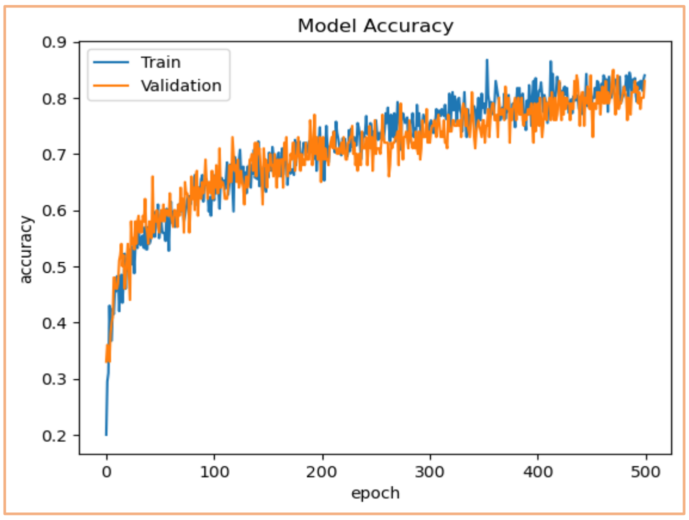
2.2.1 Model Accuracy:



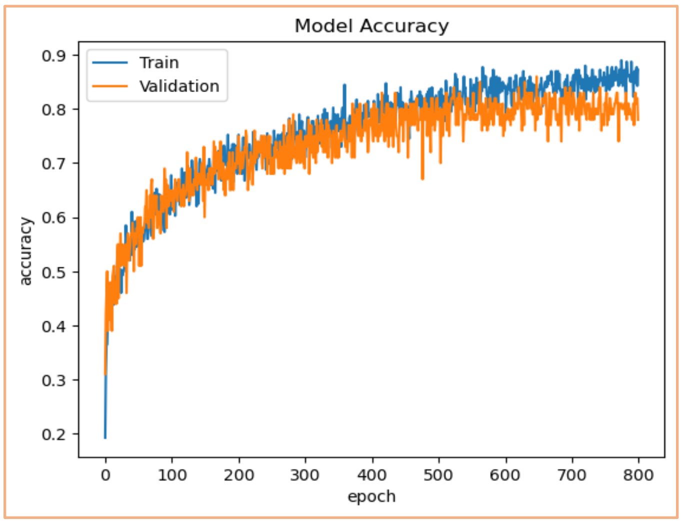
Graph 1 at 150 Epoch



Graph 2 at 300 Epoch



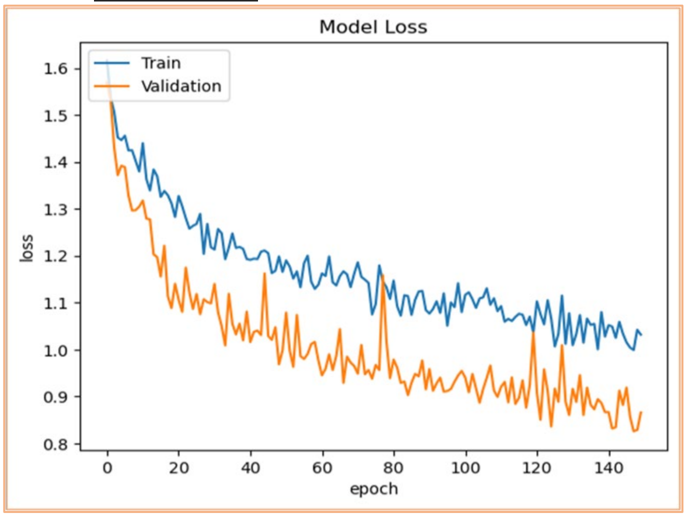
Graph 3 at 500 Epoch



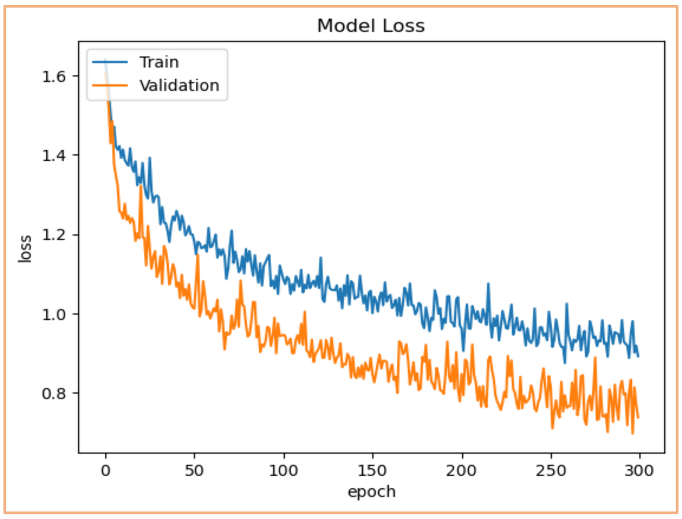
Graph 4 at 800 Epoch

The accuracy graphs show increasing trends for both training and validation set with minimal difference between them for graph 1,2,3 with increasing epoch which suggested that the model is a good fit, thus this motivates the continual increasing of epoch no. to get higher accuracy. However, at graph 4 with 800 epoch the accuracy between the training and validation set starts to diverge with training set exceeding the validation set this indicates the starting of an overfitting model. Therefore, 800 epoch is the limit of this model to have a good fit for our future prediction.

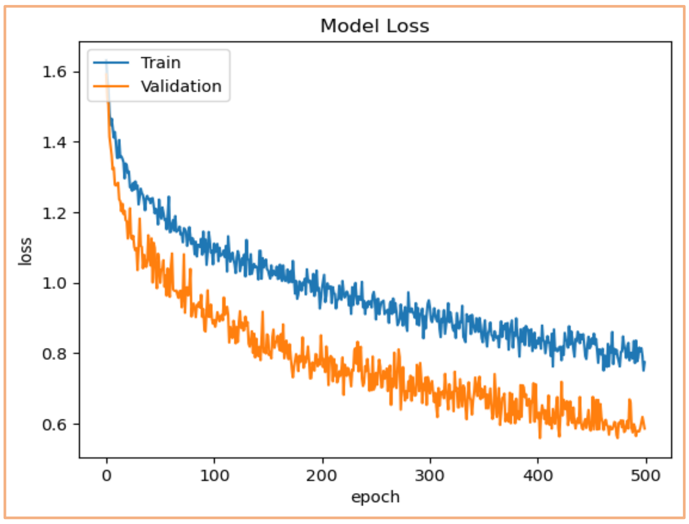
2.2.2 Model Loss



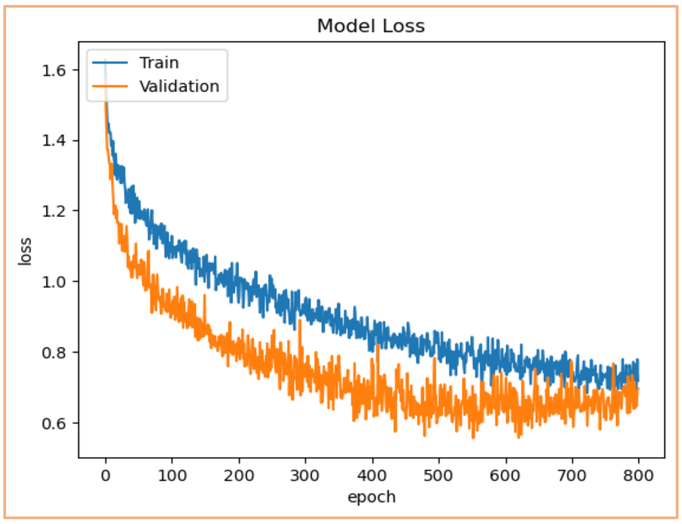
Graph 1 at 150 Epoch



Graph 2 at 300 Epoch



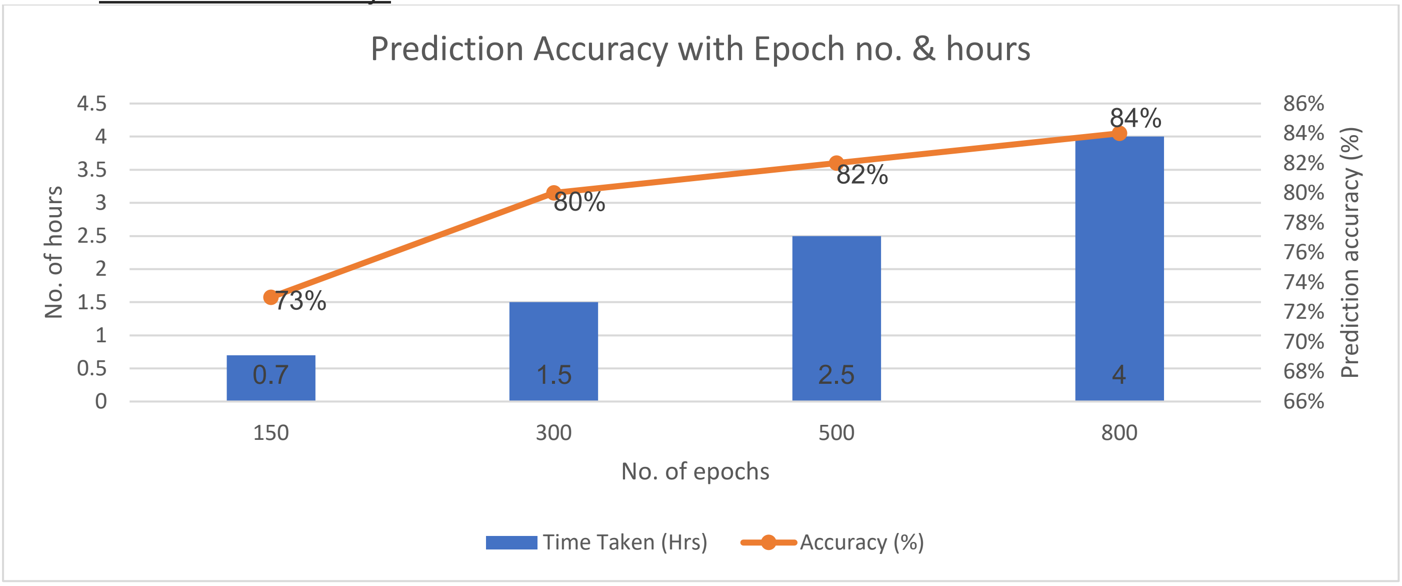
Graph 3 at 500 Epoch



Graph 4 at 800 Epoch

The graphs of 1,2,3 show that there is a decreasing trend happening for both training and validation set losses, which prompted the increasing of epoch no. as well which coincides with the accuracy graphs 1,2,3. Graph 4 at 800 epoch shows that there is a convergence happening when the validation set loss starts to increase while training set loss continue to decrease, which are starting signs of overfitting. Nevertheless, this indicates that the model is stable and is a good fit at 800 epochs as well. In addition, the reason why the validation set loss is consistently below the training set is because that our validation set mostly is made up of “easy” images to predict, as Cutmix was only done on the training set. This also shows that Cutmix brings in higher complexity into the training set than Mixup for this case. This also recommends that validation set should also have comparable complexity with the training set for future improvements to give a more accurate insights of the performance of the model.

2.2.3 Prediction Accuracy:



Graph of prediction accuracy with epoch no & hours for cutmix.

The graph above overall shows that there is an increasing trend of the prediction accuracy when there is an increase of epochs no. which indicates that increasing epochs no. increases the model prediction accuracy. However, the incremental change of the prediction accuracy is not very significant after 300 epochs which might not worth the time taken to increase the prediction accuracy by a small number in this case is 2%; 500 epochs took 2.5 hours & 800 epochs took 4 hours. Nevertheless, the final run at 800 epochs gave us the highest prediction accuracy of 84%.

2.3 Comparing between Cutmix & Mixup Prediction Accuracy

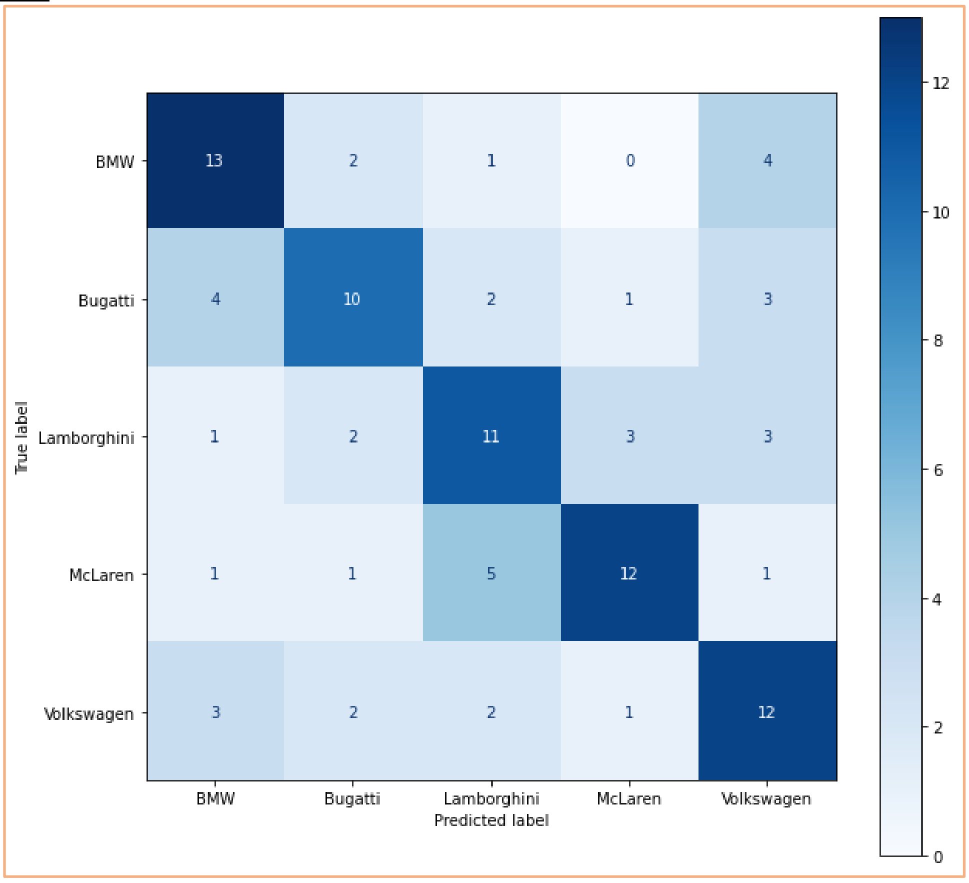
	Mixup @300 Epochs	Cutmix @ 800 Epoch
Highest Prediction Accuracy (%)	79	84

Table of Highest Prediction Accuracy between Mixup and Cutmix

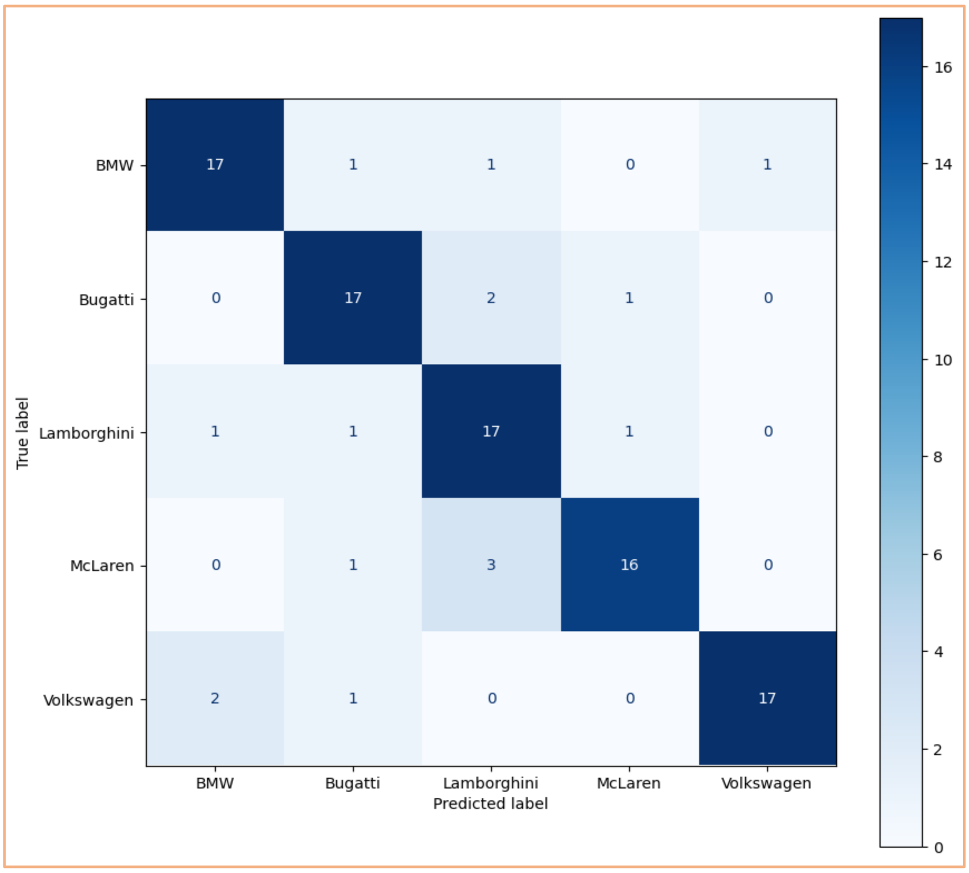
The table shows that Cutmix overall gives us the highest prediction accuracy, thus it will be implemented to our current model.

2.4 Overall Comparison between before and after new implementations (data augmentation)

Confusion matrix



Confusion matrix 1 before implementation of data augmentation



Confusion matrix 2 after implementation of data augmentation

Comparing both confusion matrixes, confusion matrix 2 has more correctly predicted classes than confusion matrix 1 that is indicated by the increase no. of darker shades boxes and its numbers have about 17 correctly predicted for each class.

Prediction accuracy

	Before implementation of data augmentation	After implementation of data augmentation
Prediction Accuracy (%)	58	84
Increase of Prediction Accuracy (%)	26	
Percentage Change (%)	48	

Table of Prediction Accuracy before and after implementation of data augmentation

The table shows that there is an increase of 26% prediction accuracy after the implementation of data augmentation that amounts to 48% percentage change.

F_1 score

Classification Report						
	Precision		Recall		F_1 Score	
	Before	After	Before	After	Before	After
BMW	0.59	0.85	0.65	0.85	0.62	0.85
Bugatti	0.59	0.81	0.50	0.85	0.54	0.83
Lamborghini	0.52	0.74	0.55	0.85	0.54	0.79
McLaren	0.71	0.89	0.60	0.80	0.65	0.84
Volkswagen	0.52	0.94	0.60	0.85	0.56	0.89
Average	0.59	0.85	0.58	0.84	0.58	0.84
Average Increase	0.26					

Table of Classification Report before & after implementation of data augmentation

The new implementation improved the model ability to correctly predict and reduce false negative in all the classes which is shown in the table with an average of 0.26 increase in precision, recall after implementation of data augmentation. This resulted in an increase average of 0.26 of the F_1 score also, with the new F_1 score being 0.84. There is still certain level of biasness, with Volkswagen and Bugatti having 0.79 and 0.89 respectively of the F_1 score.

Conclusion

Therefore, the implementation of data augmentation to regularize the model is effective with providing better generalization as shown by the improved confusion matrix, increased of prediction accuracy and F_1 score.

Reference

[1] "sklearn.metrics.classification_report," *scikit-learn*, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html.

[2] J. Brownlee, "How To Improve Deep Learning Performance - MachineLearningMastery.com," MachineLearningMastery.com, Sep. 20, 2016. [Online]. Available: https://machinelearningmastery.com/improve-deep-learning-performance/.

[3] P. varma, "Why Does Image Data Augmentation Work As A Regularizer in Deep Learning?," *Analytics India Magazine*, Sep. 02, 2020. [Online]. Available: https://analyticsindiamag.com/why-does-image-data-augmentation-work-as-a-regularizer-in-deep-learning/#:~:text=Data%20augmentation%20techniques.-,Data%20Augmentation%20As%20a%20Regularizer%20and%20Data%20Generator,a%20prediction%20with%20unseen%20data

[4] "Building powerful image classification models using very little data," *Keras.io*, 2016. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.

[5] Data augmentation, "Data augmentation | TensorFlow Core," *TensorFlow*, 2022. [Online]. Available: https://www.tensorflow.org/tutorials/images/data_augmentation.

[6] A. Kathuria, "Data Augmentation for Object Detection: How to rotate bounding boxes," *Paperspace Blog*, 09-Apr-2021. [Online]. Available: https://blog.paperspace.com/data-augmentation-for-object-detection-rotation-and-shearing/.

[7] S. Sinha, "Image Translation using OpenCV | Python," *GeeksforGeeks*, May 17, 2019. [Online]. Available: https://www.geeksforgeeks.org/image-translation-using-opencv-python/.

[8] K. Team, "Keras documentation: CutMix, MixUp, and RandAugment image augmentation with KerasCV," *Keras.io*, 2022. [Online]. Available:https://keras.io/guides/keras_cv/cut_mix_mix_up_and_rand_augment/.

[9] Saife245, "Cutmix.. vs Mixup.. vs Gridmask ..vs Cutout...," *Kaggle.com*, Jun. 2020. [Online]. Available: https://www.kaggle.com/code/saife245/cutmix-vs-mixup-vs-gridmask-vs-cutout.

[10] B. Kim, "CutMixImageDataGenerator (Keras)," *GitHub*, Feb. 11, 2020. [Online]. Available: https://github.com/devbruce/CutMixImageDataGenerator_For_Keras

[11] C. Zhang, "How to do mixup training from image files in Keras | DLology Blog," *GitHub*, Jan. 13, 2019. [Online]. Available: https://github.com/Tony607/keras_mixup_generator

[12] Data augmentation, "Data augmentation | TensorFlow Core," *TensorFlow*, 2022. [Online]. Available: https://www.tensorflow.org/tutorials/images/data_augmentation. [Accessed: Nov. 22, 2022]