Project Development Phase

Sprint 3

Team ID: PNT2022TMID35710

Project Name: Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

Image Pre Processing

```
In [ ]: from google.colab import drive
drive.mount('/content/drive')
```

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

```
In [ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
In [ ]: train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)
```

```
In [ ]: val_datagen = ImageDataGenerator(rescale = 1./255)
```

For Body

```
In [ ]: training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Car damage/body/training',
                                                 target_size=(224,224),
                                                 batch_size=10,
                                                 class_mode='categorical')

test_set = val_datagen.flow_from_directory('/content/drive/MyDrive/Car damage/body/validation',
                                            target_size =(224,224),
                                            batch_size=10,
                                            class_mode='categorical')
```

```
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
```

For the level of Damage:

```
In [ ]: training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Car damage/level/training',
                                                 target_size=(224,224),
                                                 batch_size=10,
                                                 class_mode='categorical')

test_set = val_datagen.flow_from_directory('/content/drive/MyDrive/Car damage/level/validation',
                                            target_size=(224,224),
                                            batch_size=10,
                                            class_mode='categorical')
```

```
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
```

Model Building for Body

1.Importing the Model Building Libraries

```
In [ ]: import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

Loading the Model

```
In [ ]: Image_Size = [224,224,3]
train_path = '/content/drive/MyDrive/Car damage/level/training'
valid_path = '/content/drive/MyDrive/Car damage/level/validation'
```

```
In [ ]: vgg16 = VGG16(input_shape=Image_Size, weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf
_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

Adding Flatten Layer

```
In [ ]: for layer in vgg16.layers:
  layer.trainable = False
```

```
In [ ]: folders = glob('/content/drive/MyDrive/Car damage/body/training/*')
```

```
In [ ]: folders
```

```
Out[ ]: ['/content/drive/MyDrive/Car damage/body/training/02-side',
 '/content/drive/MyDrive/Car damage/body/training/00-front',
 '/content/drive/MyDrive/Car damage/body/training/01-rear']
```

```
In [ ]: x = Flatten()(vgg16.output)
```

```
In [ ]: len(folders)
```

```
Out[ ]: 3
```

Adding Output Layer

```
In [ ]: prediction = Dense(len(folders), activation='softmax')(x)
```

Creating a Model Object

```
In [ ]: model = Model(inputs=vgg16.input, outputs=prediction)
```

```
In [ ]: model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 3)                 75267

=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

Configure the Learning Process

```
In [ ]: model.compile(
    loss = 'categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

Train the Model

```
In [ ]: r = model.fit_generator(
    training_set,
    validation_data = test_set, epochs=25, steps_per_epoch = len(training_set), validation_steps=len(test_set)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will
be removed in a future version. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/25
98/98 [==============================] - 182s 2s/step - loss: 1.4411 - accuracy: 0.5383 - val_loss: 1.1532 - val_accurac
y: 0.5965
Epoch 2/25
98/98 [==============================] - 14s 144ms/step - loss: 0.6943 - accuracy: 0.7263 - val_loss: 1.0144 - val_accura
cy: 0.6082
Epoch 3/25
98/98 [==============================] - 15s 154ms/step - loss: 0.5403 - accuracy: 0.7967 - val_loss: 1.1200 - val_accura
cy: 0.6374
Epoch 4/25
98/98 [==============================] - 14s 142ms/step - loss: 0.3849 - accuracy: 0.8519 - val_loss: 1.0220 - val_accura
cy: 0.6374
Epoch 5/25
98/98 [==============================] - 14s 144ms/step - loss: 0.3407 - accuracy: 0.8703 - val_loss: 1.4834 - val_accura
cy: 0.5497
Epoch 6/25
98/98 [==============================] - 14s 144ms/step - loss: 0.2134 - accuracy: 0.9213 - val_loss: 1.0786 - val_accura
cy: 0.6257
Epoch 7/25
98/98 [==============================] - 14s 143ms/step - loss: 0.1887 - accuracy: 0.9346 - val_loss: 1.0857 - val_accura
cy: 0.6023
Epoch 8/25
98/98 [==============================] - 14s 142ms/step - loss: 0.1564 - accuracy: 0.9438 - val_loss: 1.2026 - val_accura
cy: 0.6491
Epoch 9/25
98/98 [==============================] - 14s 143ms/step - loss: 0.1520 - accuracy: 0.9408 - val_loss: 1.1961 - val_accura
cy: 0.6374
Epoch 10/25
98/98 [==============================] - 14s 143ms/step - loss: 0.1477 - accuracy: 0.9479 - val_loss: 1.5219 - val_accura
cy: 0.6140
Epoch 11/25
98/98 [==============================] - 14s 144ms/step - loss: 0.1649 - accuracy: 0.9326 - val_loss: 1.4042 - val_accura
cy: 0.5789
Epoch 12/25
98/98 [==============================] - 14s 142ms/step - loss: 0.1156 - accuracy: 0.9581 - val_loss: 1.2789 - val_accura
cy: 0.6199
Epoch 13/25
98/98 [==============================] - 14s 142ms/step - loss: 0.0680 - accuracy: 0.9826 - val_loss: 1.3588 - val_accura
cy: 0.5965
Epoch 14/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0689 - accuracy: 0.9734 - val_loss: 1.2983 - val_accura
cy: 0.5731
Epoch 15/25
98/98 [==============================] - 15s 154ms/step - loss: 0.0632 - accuracy: 0.9857 - val_loss: 1.3064 - val_accura
cy: 0.6374
Epoch 16/25
98/98 [==============================] - 14s 146ms/step - loss: 0.0441 - accuracy: 0.9908 - val_loss: 1.2734 - val_accura
cy: 0.6374
Epoch 17/25
98/98 [==============================] - 14s 145ms/step - loss: 0.0389 - accuracy: 0.9949 - val_loss: 1.2831 - val_accura
cy: 0.6257
Epoch 18/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0476 - accuracy: 0.9918 - val_loss: 1.3047 - val_accura
cy: 0.6433
Epoch 19/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0526 - accuracy: 0.9867 - val_loss: 1.3412 - val_accura
cy: 0.6550
Epoch 20/25
98/98 [==============================] - 14s 145ms/step - loss: 0.0450 - accuracy: 0.9877 - val_loss: 1.3749 - val_accura
cy: 0.6316
Epoch 21/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0350 - accuracy: 0.9939 - val_loss: 1.3882 - val_accura
cy: 0.6550
Epoch 22/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0244 - accuracy: 0.9959 - val_loss: 1.5301 - val_accura
cy: 0.6082
Epoch 23/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0232 - accuracy: 0.9980 - val_loss: 1.5163 - val_accura
cy: 0.6374
Epoch 24/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0253 - accuracy: 0.9959 - val_loss: 1.3884 - val_accura
cy: 0.6433
Epoch 25/25
98/98 [==============================] - 14s 142ms/step - loss: 0.0131 - accuracy: 1.0000 - val_loss: 1.4228 - val_accura
cy: 0.6433
```

Save the Model

```
In [ ]: from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/Vehicle damage/body.h5')
```

Test the model

```
In [ ]: from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
```

```
In [ ]: model = load_model('/content/drive/MyDrive/Vehicle damage/body.h5')
```

```
In [ ]: def detect (frame):
  img=cv2. resize (frame, (224, 224) )
  img = cv2. cvtColor (img, cv2. COLOR_BGR2RGB)
  if (np.max (img) >1):
    img = img/255.0
  img = np.array([img])
  prediction = model .predict (img)
  label = ["front", "rear", "side"]
  preds = label[np.argmax(prediction)]
  return preds
```

```
In [ ]: import numpy as np
```

```
In [ ]: data = '/content/drive/MyDrive/Car damage/body/training/01-rear/0008.JPEG'
image= cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 0s 176ms/step
rear
```

For Level Damage

Import the ImageDataGenerator Library

```
In [ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Model Building

```
In [ ]: import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

Loading the Model

```
In [ ]: Image_Size = [224,224,3]

train_path='/content/drive/MyDrive/Car damage/level/training'
valid_path='/content/drive/MyDrive/Car damage/level/validation'
```

```
In [ ]: vgg16 = VGG16(input_shape=Image_Size, weights='imagenet', include_top=False)
```

Adding Flatten Layer

```
In [ ]: for layer in vgg16.layers:
  layer.trainable = False
```

```
In [ ]: folders = glob('/content/drive/MyDrive/Car damage/level/training/*')
```

```
In [ ]: folders
```

```
Out[ ]: ['/content/drive/MyDrive/Car damage/level/training/02-moderate',
 '/content/drive/MyDrive/Car damage/level/training/03-severe',
 '/content/drive/MyDrive/Car damage/level/training/01-minor']
```

```
In [ ]: x = Flatten() (vgg16.output)
```

```
In [ ]: len(folders)
```

```
Out[ ]: 3
```

Adding Output Layers

```
In [ ]: prediction = Dense(len(folders), activation='softmax')(x)
```

Creating a model Object

```
In [ ]: model = Model(inputs=vgg16.input, outputs=prediction)
```

```
In [ ]: model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_1 (Dense)             (None, 3)                 75267

=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

Configure the learining Process

```
In [ ]: model.compile(
loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy']
)
```

Train the Model

```
In [ ]: r = model.fit_generator(
    training_set,
    validation_data = test_set, epochs=25, steps_per_epoch = len(training_set), validation_steps=len(test_set)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will
be removed in a future version. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/25
98/98 [==============================] - 16s 159ms/step - loss: 1.0755 - accuracy: 0.5689 - val_loss: 0.8828 - val_accura
cy: 0.6433
Epoch 2/25
98/98 [==============================] - 14s 146ms/step - loss: 0.7613 - accuracy: 0.7058 - val_loss: 0.9603 - val_accura
cy: 0.6550
Epoch 3/25
98/98 [==============================] - 14s 143ms/step - loss: 0.6753 - accuracy: 0.7538 - val_loss: 1.6167 - val_accura
cy: 0.5965
Epoch 4/25
98/98 [==============================] - 14s 144ms/step - loss: 0.3768 - accuracy: 0.8519 - val_loss: 1.1048 - val_accura
cy: 0.6082
Epoch 5/25
98/98 [==============================] - 14s 145ms/step - loss: 0.2910 - accuracy: 0.8958 - val_loss: 1.1118 - val_accura
cy: 0.6374
Epoch 6/25
98/98 [==============================] - 14s 146ms/step - loss: 0.2471 - accuracy: 0.9101 - val_loss: 1.1026 - val_accura
cy: 0.6140
Epoch 7/25
98/98 [==============================] - 14s 144ms/step - loss: 0.1944 - accuracy: 0.9387 - val_loss: 1.3009 - val_accura
cy: 0.6023
Epoch 8/25
98/98 [==============================] - 14s 144ms/step - loss: 0.1994 - accuracy: 0.9285 - val_loss: 1.3112 - val_accura
cy: 0.5673
Epoch 9/25
98/98 [==============================] - 14s 143ms/step - loss: 0.1496 - accuracy: 0.9520 - val_loss: 1.1786 - val_accura
cy: 0.6374
Epoch 10/25
98/98 [==============================] - 14s 143ms/step - loss: 0.1142 - accuracy: 0.9622 - val_loss: 1.1918 - val_accura
cy: 0.6433
Epoch 11/25
98/98 [==============================] - 14s 145ms/step - loss: 0.0788 - accuracy: 0.9785 - val_loss: 1.1670 - val_accura
cy: 0.6550
Epoch 12/25
98/98 [==============================] - 15s 153ms/step - loss: 0.0730 - accuracy: 0.9826 - val_loss: 1.3358 - val_accura
cy: 0.6082
Epoch 13/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0867 - accuracy: 0.9755 - val_loss: 1.1960 - val_accura
cy: 0.6316
Epoch 14/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0908 - accuracy: 0.9714 - val_loss: 1.2994 - val_accura
cy: 0.6374
Epoch 15/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0538 - accuracy: 0.9847 - val_loss: 1.4916 - val_accura
cy: 0.5848
Epoch 16/25
98/98 [==============================] - 14s 146ms/step - loss: 0.0539 - accuracy: 0.9857 - val_loss: 1.6840 - val_accura
cy: 0.6433
Epoch 17/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0506 - accuracy: 0.9888 - val_loss: 1.3116 - val_accura
cy: 0.6491
Epoch 18/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0387 - accuracy: 0.9908 - val_loss: 1.5104 - val_accura
cy: 0.5906
Epoch 19/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0339 - accuracy: 0.9928 - val_loss: 1.4485 - val_accura
cy: 0.6082
Epoch 20/25
98/98 [==============================] - 14s 146ms/step - loss: 0.0304 - accuracy: 0.9980 - val_loss: 1.4151 - val_accura
cy: 0.6257
Epoch 21/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0176 - accuracy: 1.0000 - val_loss: 1.3324 - val_accura
cy: 0.6550
Epoch 22/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0653 - accuracy: 0.9806 - val_loss: 1.5970 - val_accura
cy: 0.6082
Epoch 23/25
98/98 [==============================] - 14s 144ms/step - loss: 0.0717 - accuracy: 0.9765 - val_loss: 1.9383 - val_accura
cy: 0.5848
Epoch 24/25
98/98 [==============================] - 15s 155ms/step - loss: 0.0529 - accuracy: 0.9826 - val_loss: 1.4523 - val_accura
cy: 0.6199
Epoch 25/25
98/98 [==============================] - 14s 143ms/step - loss: 0.0612 - accuracy: 0.9765 - val_loss: 1.5688 - val_accura
cy: 0.6433
```

Save the Model

```
In [ ]: from tensorflow.keras.models import load_model
model.save('/content/damage vehicle/Model/level.h5')
```

Test the Model

```
In [ ]: from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
```

```
In [ ]: model = load_model('/content/damage vehicle/Model/level.h5')
```

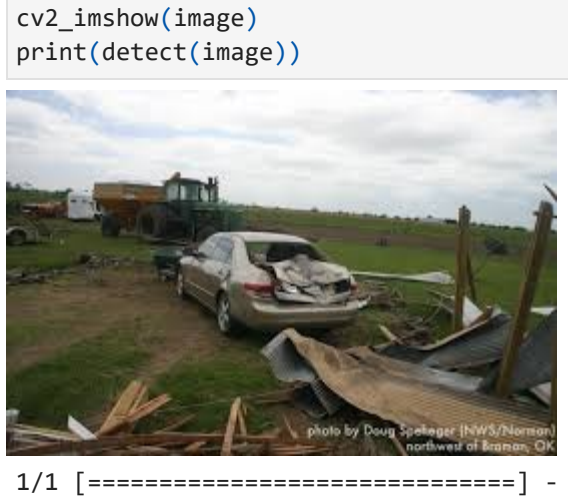
```
In [ ]: def detect(frame):
  img = cv2.resize (frame, (224,224))
  img = cv2.cvtColor (img, cv2.COLOR_BGR2RGB)

  if (np.max(img)>1):
    img = img/255.0
  img = np.array([img])
  prediction = model.predict (img)
  label = ["minor", "moderate", "severe"]
  preds = label[np.argmax (prediction) ]
  return preds
```

```
In [ ]: import numpy as np
import PIL
from IPython import display

def cv2_imshow(a):
    a = a.clip(0, 255).astype('uint8')
    if a.ndim == 3:
        if a.shape[2] == 4:
            a = cv2.cvtColor(a, cv2.COLOR_BGRA2RGBA)
        else:
            a = cv2.cvtColor(a, cv2.COLOR_BGR2RGB)
    display.display(PIL.Image.fromarray(a))
```

```
In [ ]: data = '/content/drive/MyDrive/Car damage/level/validation/03-severe/0004.JPEG'
image = cv2.imread(data)
cv2_imshow(image)
print(detect(image))
```



```
1/1 [==============================] - 0s 118ms/step
severe
```