METHODOLOGY

The project was completed in a series of steps which begin with the libraries and dataset import, data preprocessing, model initialization and end the evaluation of the model to obtain the test accuracy. Libraries such as NumPy, pandas, matplotlib, TensorFlow and Keras are imported. The dataset is then imported. Moving on, the data preprocessing was carried out first setting the image size to 256x256 and initializing the labels as given in the dataset. All the train and test images are then converted into grayscale and standardized.

a. Convolutional Neural Network

The first approach that we used was a Convolutional neural network. Convolutional neural networks use convolution for the purpose of extraction of features. This algorithm uses smaller filters to reduce the size of the image and give a prediction.

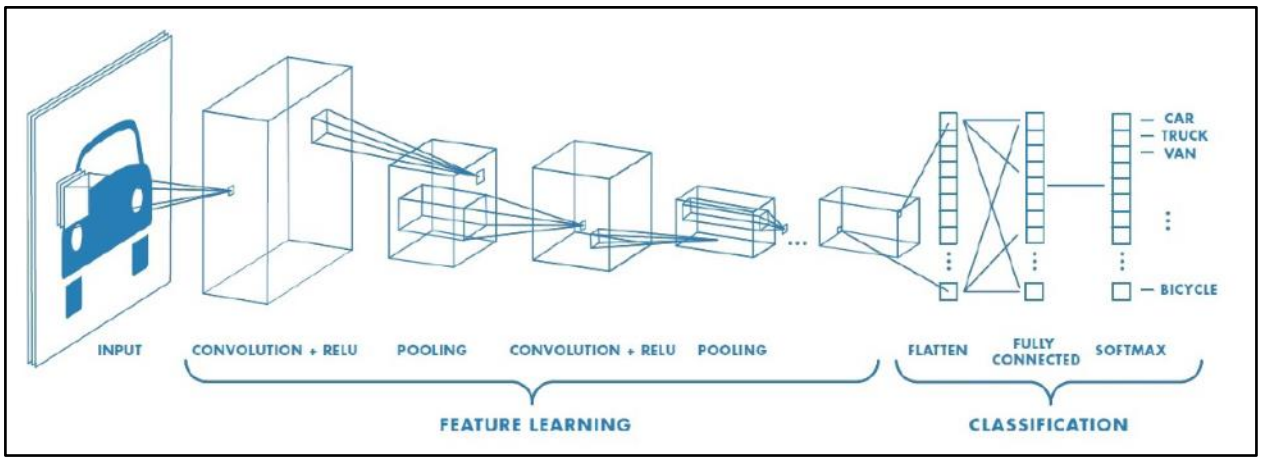


Figure 1: General form of a convolutional neural network

A general convolutional neural network deploys three stages for the purpose of classification. They are convolutional, detector and pooling stages. The convolutional stage uses linear activation functions. The detector stage uses non-linear activation functions such as rectified linear unit (ReLU). The max-pooling layer takes the output of all layers are provides an output based on the confidence score. Basically, pooling layer modifies the output with a summary statistic of nearby outputs.

```
In [22]: model = Sequential()

In [23]: model.add(Conv2D(32, (3,3), 1, activation='relu', input_shape=(256,256,3)))
         model.add(MaxPooling2D())
         model.add(Conv2D(64, (3,3), 1, activation='relu'))
         model.add(MaxPooling2D())
         model.add(Conv2D(32, (3,3), 1, activation='relu'))
         model.add(MaxPooling2D())
         model.add(Flatten())
         model.add(Dense(256, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))

In [24]: model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

Figure 2: Our model of CNN

The model used in this project uses three convolutional layers with 32, 64 and 32 neurons respectively with a rectified linear unit activation function. Three max-pooling and two dense layers with ReLU and sigmoid activation functions were applied to the neural network. The model was optimized using the adam optimizer and the binary cross entropy loss function. The summary of the model can be seen in figure 3.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 63, 63, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 61, 61, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 256)               7872768

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 7,910,881
Trainable params: 7,910,881
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: CNN model summary

After the model is developed, the epochs are set to 100 and the batch size is initialized to 128. Further, the model is executed to obtain the train, validation and test accuracy.

b. Residual Neural Network (ResNet50)

A residual neural network is a combination of a convolutional neural networks with multiple convolutional layers and skips connections.

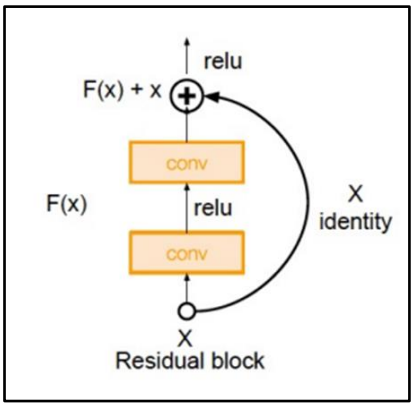


Figure 4: Residual block showing skip connection

When the number of convolutional layers is stacked on top of one another, a deep neural network is created. This increases the features of the model. But, on the other hand, a problem of degradation is introduced. Due to this issue, the accuracy of the model degrades after a certain threshold. A residual block has three convolution layers followed by a batch normalization layer and a rectified linear activation function. This is again continued by three convolution layers and a batch normalization layer. The skip connection skips both these layers and adds directly before the ReLU activation function. Such residual blocks are repeated to form a residual network.

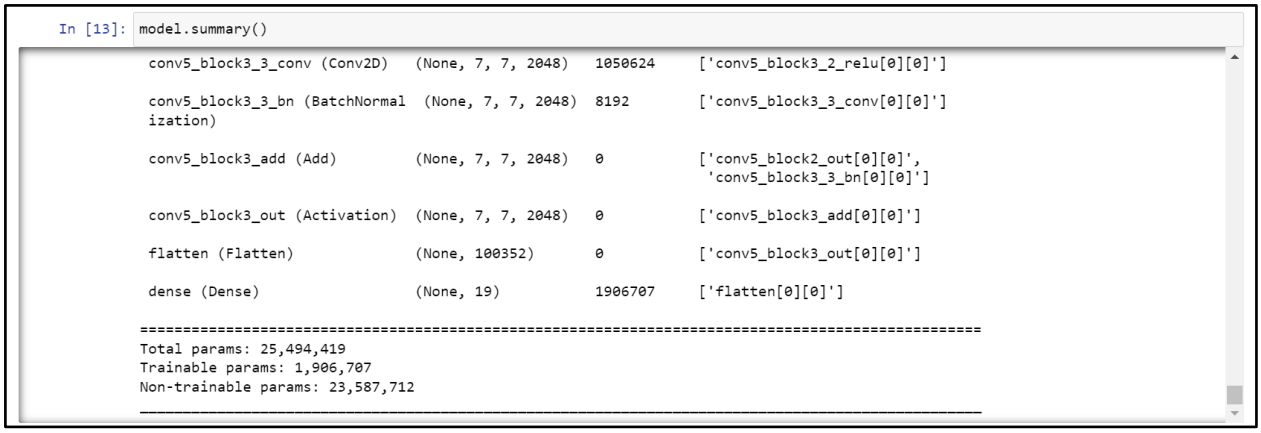
```
In [7]:
for layer in resnet.layers:
    layer.trainable = False

# This will let us use the default weights used by the imagenet.
```

```
In [5]:
resnet = ResNet50(
    input_shape = IMAGE_SIZE + [3], # Making the image into 3 Channel, so concating 3.
    weights = 'imagenet', # Default weights.
    include_top = False   #
)
```

Figure 5: ResNet50 model import

ResNet50 supports an image size of 224x224. In our project, we have imported a defined and initialized model. The default imagenet weights are used in this algorithm.



```
In [13]: model.summary()
```

```
 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5_block2_out[0][0]',
                                                                  'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5_block3_add[0][0]']

 flatten (Flatten)              (None, 100352)       0           ['conv5_block3_out[0][0]']

 dense (Dense)                  (None, 19)           1906707     ['flatten[0][0]']

==================================================================================================
Total params: 25,494,419
Trainable params: 1,906,707
Non-trainable params: 23,587,712
__________________________________________________________________________________________________
```

Figure 6: ResNet50 Model summary

The 50-layer ResNet model consists of 48 convolutional layers, one max pooling layer and one average pooling layer. The model was compiled using the Adam optimizer and categorical cross-entropy loss function. The epochs and batch size were set to 10 and 32 respectively. Further, the train, validation and test accuracies were obtained. To show the performance of the model in terms of test data, we have plotted the confusion matrix and converted it into a comma-separated file.