importing libraries

In [ ]:
```
# Importing libraries
import tensorflow as tf # deeplearning lib
import matplotlib.pyplot as plt # visualization
import numpy as np # arrays and matrices
import cv2 # computer vision
import os # communicate with system
import imghdr # handling image formats
```

```
C:\Users\mahig\AppData\Local\Temp\ipykernel_18720\877037246.py:7: DeprecationWarning: 'imghdr' is deprecated an
d slated for removal in Python 3.13
  import imghdr # handling image formats
```

Importing Data

In [ ]:
```
# base directory for dataset
data_dir = 'data'
```

In [ ]:
```
# desired image formats
img_ext = ['jpeg','jpg','bmp','png']
```

In [ ]:
```
# getting the data directories inside base directory
os.listdir(data_dir)
```

Out[ ]:
```
['bottle', 'fishnet', 'metal_debris', 'plastic_bag', 'tyre']
```

Data Preparation

In [ ]:
```
# getting the directories inside the data folder as list
for img_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir,img_class)): # iterate through every images in each folder
        img_path = os.path.join(data_dir,img_class,image) # file path for each images
        try:
            tip = imghdr.what(img_path)
            if tip not in img_ext: # checks for desired extensions
                print(f'Oopss:( image not found in ext list - {img_path}')
                os.remove(img_path) # removes the file from device
        except Exception as e:
             print(f'Issue with image {img_path}: {e}')
```

In [ ]:
```
# getting number of images in each folders
print('bottle -',len(os.listdir(os.path.join(data_dir,'bottle'))))
print('fishnet -',len(os.listdir(os.path.join(data_dir,'fishnet'))))
print('metal_debris',len(os.listdir(os.path.join(data_dir,'metal_debris'))))
print('plastic_bag -',len(os.listdir(os.path.join(data_dir,'plastic_bag'))))
print('tyre -',len(os.listdir(os.path.join(data_dir,'tyre'))))
```

```
bottle - 273
fishnet - 254
metal_debris 223
plastic_bag - 229
tyre - 214
```

Importing data

In [ ]:
```
# creates a dataset from images in directory
data=tf.keras.utils.image_dataset_from_directory('data')
```

```
Found 1193 files belonging to 5 classes.
```

In [ ]:
```
data
```

Out[ ]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(
shape=(None,), dtype=tf.int32, name=None))>
```

In [ ]:
```
# represent data as numpy array
data_iterator=data.as_numpy_iterator()
```

In [ ]:
```
data_iterator.next()
```

Out[ ]:
```
(array([[[[ 27.90625  ,  52.90625  ,  57.90625  ],
          [ 26.       ,  51.       ,  56.       ],
          [ 24.09375  ,  51.09375  ,  60.28125  ],
          ...,
          [ 20.       ,  34.       ,  37.       ],
          [ 22.       ,  33.       ,  37.       ],
```

```
  [ 19.90625  ,  35.1875   ,  38.09375  ]],

 [[ 27.       ,  55.4375   ,  63.15625  ],
  [ 26.       ,  54.4375   ,  62.15625  ],
  [ 26.28125  ,  56.71875  ,  60.       ],
  ...,
  [ 20.       ,  34.       ,  37.       ],
  [ 18.28125  ,  33.       ,  36.       ],
  [ 19.       ,  33.28125  ,  36.28125  ]],

 [[ 28.       ,  58.       ,  66.       ],
  [ 27.       ,  57.       ,  65.       ],
  [ 27.0625   ,  58.125    ,  68.65625  ],
  ...,
  [ 20.       ,  34.       ,  37.       ],
  [ 16.       ,  34.       ,  36.       ],
  [ 18.0625   ,  30.46875  ,  34.       ]],

 ...,

 [[ 34.53125  ,  97.53125  , 112.53125  ],
  [ 33.53125  ,  92.53125  , 108.53125  ],
  [ 30.53125  ,  87.53125  , 104.53125  ],
  ...,
  [ 20.       ,  47.       ,  56.       ],
  [ 20.46875  ,  42.46875  ,  53.46875  ],
  [ 22.       ,  44.       ,  55.       ]],

 [[ 31.4375   ,  89.875    , 106.15625  ],
  [ 30.       ,  86.71875  , 103.71875  ],
  [ 29.       ,  86.       , 103.       ],
  ...,
  [ 19.4375   ,  47.       ,  55.71875  ],
  [ 22.71875  ,  45.28125  ,  56.       ],
  [ 24.       ,  46.5625   ,  57.28125  ]],

 [[ 29.09375  ,  84.8125   , 101.90625  ],
  [ 31.       ,  82.       , 101.       ],
  [ 31.       ,  88.       , 105.       ],
  ...,
  [ 17.       ,  47.       ,  55.       ],
  [ 19.90625  ,  46.90625  ,  55.90625  ],
  [ 20.90625  ,  47.90625  ,  56.90625  ]]],


[[[ 38.       , 113.171875 , 132.08594  ],
  [ 38.515625 , 115.515625 , 133.       ],
  [ 39.570312 , 118.71875  , 133.85938  ],
  ...,
  [ 32.289062 ,  96.28906  , 106.28906  ],
  [ 28.773438 ,  92.77344  , 102.77344  ],
  [ 25.257812 ,  89.25781  ,  99.25781  ]],

 [[ 38.44275  , 113.614624 , 132.52869  ],
  [ 38.515625 , 115.515625 , 133.       ],
  [ 39.778442 , 118.92688  , 134.0675   ],
  ...,
  [ 33.534058 ,  97.53406  , 107.53406  ],
  [ 29.867065 ,  93.867065 , 103.867065 ],
  [ 26.669312 ,  90.66931  , 100.66931  ]],

 [[ 39.066772 , 114.36719  , 132.87146  ],
  [ 38.796875 , 115.796875 , 133.07251  ],
  [ 40.220825 , 119.36926  , 134.50989  ],
  ...,
  [ 35.060425 ,  99.4823   , 109.261475 ],
  [ 31.276245 ,  95.69812  , 105.41687  ],
  [ 28.324585 ,  92.74646  , 102.46521  ]],

 ...,

 [[ 45.197998 ,  75.97217  ,  60.16748  ],
  [ 43.17981  ,  77.070435 ,  60.38794  ],
  [ 37.872192 ,  77.24927  ,  60.550293 ],
  ...,
  [ 26.140625 ,  52.140625 ,  43.140625 ],
  [ 25.398438 ,  50.65625  ,  43.882812 ],
  [ 25.140625 ,  50.140625 ,  44.140625 ]],

 [[ 49.47351  ,  80.645386 ,  64.55945  ],
  [ 42.173096 ,  76.470215 ,  59.33728  ],
  [ 33.800293 ,  73.96655  ,  56.22937  ],
  ...,
  [ 25.705933 ,  51.705933 ,  42.705933 ],
  [ 25.124878 ,  50.38269  ,  43.609253 ],
  [ 24.528687 ,  49.528687 ,  43.528687 ]],

 [[ 55.3125   ,  86.484375 ,  70.39844  ],
  [ 44.390625 ,  78.9375   ,  61.679688 ],
```

```
  [ 40.445312 ,  81.16406  ,  62.734375 ],
  ...,
  [ 25.429688 ,  51.429688 ,  42.429688 ],
  [ 25.       ,  50.257812 ,  43.484375 ],
  [ 24.085938 ,  49.085938 ,  43.085938 ]]],


[[[ 35.95996  , 129.11328  , 199.11328  ],
  [ 36.5      , 129.5      , 199.5      ],
  [ 40.61328  , 131.61328  , 201.61328  ],
  ...,
  [ 23.19336  , 139.65332  , 222.5      ],
  [ 21.19336  , 141.34668  , 226.88672  ],
  [ 22.80664  , 142.80664  , 228.80664  ]],

 [[ 33.91992  , 130.91992  , 199.91992  ],
  [ 36.219727 , 133.21973  , 202.21973  ],
  [ 34.54004  , 131.37988  , 200.2998   ],
  ...,
  [ 19.240234 , 142.24023  , 222.24023  ],
  [ 22.95996  , 142.12012  , 226.04004  ],
  [ 22.379883 , 141.54004  , 225.45996  ]],

 [[ 38.466797 , 131.4668   , 201.4668   ],
  [ 37.2334   , 130.2334   , 200.2334   ],
  [ 36.533203 , 129.       , 200.59961  ],
  ...,
  [ 23.09961  , 140.5      , 223.0332   ],
  [ 23.533203 , 140.9336   , 223.4668   ],
  [ 24.566406 , 141.9668   , 224.5      ]],

 ...,

 [[193.33398  , 181.40039  , 205.2002   ],
  [133.19824  , 139.26465  , 158.26465  ],
  [126.93262  , 128.63184  , 183.33105  ],
  ...,
  [  1.5332031,  21.5      ,  34.966797 ],
  [  0.       ,  23.466797 ,  27.466797 ],
  [  0.       ,  23.533203 ,  27.533203 ]],

 [[162.67969  , 153.67969  , 148.23926  ],
  [131.       , 125.259766 , 123.55957  ],
  [150.37988  , 138.95996  , 143.59961  ],
  ...,
  [  0.6201172,  22.780273 ,  52.060547 ],
  [  2.5800781,  27.080078 ,  31.080078 ],
  [  0.       ,  24.080078 ,  28.080078 ]],

 [[158.16016  , 151.66016  , 122.66016  ],
  [190.38672  , 183.42676  , 168.27344  ],
  [167.14648  , 184.80664  , 193.4668   ],
  ...,
  [ 77.541016 , 113.774414 , 151.04102  ],
  [ 10.693359 ,  36.7334   ,  40.38672  ],
  [  1.9599609,  29.04004  ,  32.34668  ]]],


...,


[[[ 49.08008  ,  86.08008  ,  95.08008  ],
  [ 44.617188 ,  81.61719  ,  90.61719  ],
  [ 46.5      ,  83.5      ,  92.5      ],
  ...,
  [100.82031  , 156.82031  , 173.82031  ],
  [ 99.796875 , 155.79688  , 172.79688  ],
  [105.88281  , 161.88281  , 178.88281  ]],

 [[ 39.648438 ,  76.64844  ,  85.64844  ],
  [ 47.46289  ,  84.46289  ,  93.46289  ],
  [ 46.57422  ,  83.57422  ,  92.57422  ],
  ...,
  [ 93.18555  , 149.18555  , 166.18555  ],
  [114.44336  , 170.44336  , 187.44336  ],
  [ 97.166016 , 153.16602  , 170.16602  ]],

 [[ 46.95703  ,  83.95703  ,  92.95703  ],
  [ 42.40039  ,  79.40039  ,  88.40039  ],
  [ 47.5      ,  84.5      ,  93.5      ],
  ...,
  [ 91.       , 147.       , 164.       ],
  [ 96.94336  , 152.94336  , 169.94336  ],
  [ 97.51367  , 153.51367  , 170.51367  ]],

 ...,

 [[ 10.       ,  19.314453 ,  10.542969 ],
  [ 13.228516 ,  20.228516 ,  12.228516 ],
```

```
  [ 18.771484 ,  24.314453 ,  17.042969 ],
  ...,
  [  1.       ,   7.       ,   5.       ],
  [  0.       ,   6.       ,   4.       ],
  [  0.       ,   6.       ,   4.       ]],

 [[ 11.925781 ,  21.925781 ,  12.925781 ],
  [ 11.111328 ,  19.5      ,  11.037109 ],
  [ 14.6484375,  20.648438 ,  13.1484375],
  ...,
  [  1.       ,   7.       ,   5.       ],
  [  0.       ,   6.       ,   4.       ],
  [  0.       ,   6.       ,   4.       ]],

 [[ 16.5      ,  24.5      ,  13.5      ],
  [ 11.5      ,  18.5      ,  10.5      ],
  [ 15.       ,  22.       ,  14.5      ],
  ...,
  [  1.       ,   7.       ,   5.       ],
  [  0.       ,   6.       ,   4.       ],
  [  0.       ,   6.       ,   4.       ]]],


[[[155.       , 159.       , 144.       ],
  [155.       , 159.       , 144.       ],
  [155.       , 159.       , 144.       ],
  ...,
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ]],

 [[155.       , 159.       , 144.       ],
  [155.       , 159.       , 144.       ],
  [155.       , 159.       , 144.       ],
  ...,
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ]],

 [[155.14062  , 159.14062  , 143.85938  ],
  [155.14062  , 159.14062  , 143.85938  ],
  [155.14062  , 159.14062  , 143.85938  ],
  ...,
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ],
  [191.       , 193.       , 182.       ]],

 ...,

 [[  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  ...,
  [ 37.28125  ,  39.28125  ,  26.28125  ],
  [ 37.28125  ,  39.28125  ,  26.28125  ],
  [ 37.28125  ,  39.28125  ,  26.28125  ]],

 [[  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  ...,
  [ 40.09375  ,  42.09375  ,  29.09375  ],
  [ 40.09375  ,  42.09375  ,  29.09375  ],
  [ 40.09375  ,  42.09375  ,  29.09375  ]],

 [[  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  [  5.       ,   6.       ,   1.       ],
  ...,
  [ 43.       ,  45.       ,  32.       ],
  [ 43.       ,  45.       ,  32.       ],
  [ 43.       ,  45.       ,  32.       ]]],


[[[ 86.       , 137.       , 184.       ],
  [ 86.       , 137.       , 184.       ],
  [ 86.       , 137.       , 184.       ],
  ...,
  [105.18555  , 153.18555  , 191.18555  ],
  [105.       , 153.       , 191.       ],
  [105.       , 153.       , 191.       ]],

 [[ 86.572266 , 137.57227  , 184.57227  ],
  [ 86.572266 , 137.57227  , 184.57227  ],
  [ 86.572266 , 137.57227  , 184.57227  ],
  ...,
  [105.18555  , 153.18555  , 191.18555  ],
  [105.       , 153.       , 191.       ],
  [105.       , 153.       , 191.       ]],
```

```
          [[ 87.       , 138.       , 185.       ],
           [ 87.       , 138.       , 185.       ],
           [ 87.       , 138.       , 185.       ],
           ...,
           [105.18555  , 153.18555  , 191.18555  ],
           [105.       , 153.       , 191.       ],
           [105.       , 153.       , 191.       ]],

          ...,

          [[ 50.38492  ,  60.607574 ,  55.08112  ],
           [ 23.638603 ,  36.0571   ,  32.2153   ],
           [ 14.216568 ,  28.86979  ,  29.192593 ],
           ...,
           [160.58571  , 169.63898  , 165.6404   ],
           [166.81111  , 175.81111  , 172.74718  ],
           [156.63834  , 165.63834  , 162.63834  ]],

          [[ 21.914867 ,  27.551338 ,  24.26764  ],
           [ 12.04631  ,  19.76878  ,  17.021942 ],
           [ 22.043411 ,  34.756115 ,  35.661125 ],
           ...,
           [162.02591  , 171.02591  , 167.65482  ],
           [167.91956  , 176.91956  , 173.91956  ],
           [161.29321  , 170.29321  , 167.29321  ]],

          [[  0.       ,   1.0371094,   0.       ],
           [  2.8945312,   5.7851562,   4.1191406],
           [ 28.226562 ,  38.96875  ,  40.339844 ],
           ...,
           [156.51562  , 165.51562  , 162.14453  ],
           [157.11133  , 166.11133  , 163.11133  ],
           [149.29688  , 158.29688  , 155.29688  ]]]], dtype=float32),
 array([3, 3, 3, 1, 0, 4, 4, 4, 1, 0, 2, 3, 1, 1, 0, 2, 0, 4, 3, 0, 2, 4,
        3, 1, 2, 1, 3, 0, 1, 2, 3, 2]))
```

In [ ]:
```
# getting a batch from the iterator
batch = data_iterator.next()
```

In [ ]:
```
batch
```

Out[ ]:
```
(array([[[[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          ...,
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]],

         [[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          ...,
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]],

         [[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          ...,
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
          [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]],

         ...,

         [[0.00000000e+00, 1.56250000e-02, 0.00000000e+00],
          [0.00000000e+00, 1.56250000e-02, 0.00000000e+00],
          [0.00000000e+00, 1.56250000e-02, 0.00000000e+00],
          ...,
          [0.00000000e+00, 1.56250000e-02, 0.00000000e+00],
          [0.00000000e+00, 1.56250000e-02, 0.00000000e+00],
          [0.00000000e+00, 1.56250000e-02, 0.00000000e+00]],

         [[3.39062500e+00, 0.00000000e+00, 0.00000000e+00],
          [3.39062500e+00, 0.00000000e+00, 0.00000000e+00],
          [3.39062500e+00, 0.00000000e+00, 0.00000000e+00],
          ...,
          [3.39062500e+00, 0.00000000e+00, 0.00000000e+00],
          [3.39062500e+00, 0.00000000e+00, 0.00000000e+00],
          [3.39062500e+00, 0.00000000e+00, 0.00000000e+00]],

         [[5.59375000e+00, 0.00000000e+00, 0.00000000e+00],
          [5.59375000e+00, 0.00000000e+00, 0.00000000e+00],
          [5.59375000e+00, 0.00000000e+00, 0.00000000e+00],
          ...,
```

```
  [5.59375000e+00, 0.00000000e+00, 0.00000000e+00],
  [5.59375000e+00, 0.00000000e+00, 0.00000000e+00],
  [5.59375000e+00, 0.00000000e+00, 0.00000000e+00]]],


[[[8.30312500e+01, 1.50031250e+02, 1.41031250e+02],
  [8.49531250e+01, 1.52000000e+02, 1.42906250e+02],
  [8.36875000e+01, 1.51687500e+02, 1.40531250e+02],
  ...,
  [5.92187500e+00, 8.10000000e+01, 1.04078125e+02],
  [6.95312500e+00, 7.90937500e+01, 1.03046875e+02],
  [1.09375000e+01, 8.19531250e+01, 1.03984375e+02]],

 [[7.61239624e+01, 1.43123962e+02, 1.34123962e+02],
  [7.18350830e+01, 1.38881958e+02, 1.29788208e+02],
  [6.86926880e+01, 1.36595947e+02, 1.25488068e+02],
  ...,
  [5.02218628e+00, 8.20061035e+01, 1.05035858e+02],
  [6.30496216e+00, 8.03030090e+01, 1.03666016e+02],
  [1.09278259e+01, 8.44103394e+01, 1.05822449e+02]],

 [[6.93754272e+01, 1.36375427e+02, 1.27375427e+02],
  [6.50154114e+01, 1.32028046e+02, 1.22951416e+02],
  [6.37374268e+01, 1.30907776e+02, 1.20166351e+02],
  ...,
  [4.83398438e+00, 8.55468750e+01, 1.07787872e+02],
  [3.83468628e+00, 7.90951538e+01, 1.01364685e+02],
  [6.93283081e+00, 7.68994446e+01, 9.79192810e+01]],

 ...,

 [[3.69229126e+00, 1.17106354e+02, 1.63190186e+02],
  [6.47311401e+00, 1.17048584e+02, 1.63538818e+02],
  [1.24524231e+01, 1.25200470e+02, 1.72643829e+02],
  ...,
  [1.08769531e+01, 1.17355469e+02, 1.71873474e+02],
  [1.40942993e+01, 1.22287201e+02, 1.77522491e+02],
  [7.30227661e+00, 1.15571808e+02, 1.70841339e+02]],

 [[4.14184570e+00, 1.16481781e+02, 1.68147797e+02],
  [1.40597534e+00, 1.11312805e+02, 1.63007843e+02],
  [7.52816772e+00, 1.20229340e+02, 1.72544250e+02],
  ...,
  [1.31375427e+01, 1.20802521e+02, 1.77135010e+02],
  [8.54153442e+00, 1.18209503e+02, 1.74227356e+02],
  [5.42700195e+00, 1.15188721e+02, 1.71188721e+02]],

 [[1.08750000e+01, 1.23843750e+02, 1.79843750e+02],
  [3.23437500e+00, 1.14234375e+02, 1.70234375e+02],
  [8.23437500e+00, 1.19078125e+02, 1.75156250e+02],
  ...,
  [1.38437500e+01, 1.22843750e+02, 1.79843750e+02],
  [4.46875000e+00, 1.15375000e+02, 1.71421875e+02],
  [5.96875000e+00, 1.16968750e+02, 1.72968750e+02]]],


[[[1.78000000e+02, 2.25000000e+02, 2.53000000e+02],
  [1.77828125e+02, 2.24828125e+02, 2.52828125e+02],
  [1.77046875e+02, 2.24046875e+02, 2.52046875e+02],
  ...,
  [1.44757812e+02, 1.87757812e+02, 2.19757812e+02],
  [1.49054688e+02, 1.92054688e+02, 2.24054688e+02],
  [1.50000000e+02, 1.93000000e+02, 2.25000000e+02]],

 [[1.78171875e+02, 2.25171875e+02, 2.53171875e+02],
  [1.78000000e+02, 2.25000000e+02, 2.53000000e+02],
  [1.77218750e+02, 2.24218750e+02, 2.52218750e+02],
  ...,
  [1.45298279e+02, 1.88216370e+02, 2.20216370e+02],
  [1.49293030e+02, 1.92278259e+02, 2.24278259e+02],
  [1.50171875e+02, 1.93171875e+02, 2.25171875e+02]],

 [[1.78953125e+02, 2.25953125e+02, 2.53953125e+02],
  [1.78781250e+02, 2.25781250e+02, 2.53781250e+02],
  [1.78000000e+02, 2.25000000e+02, 2.53000000e+02],
  ...,
  [1.47754944e+02, 1.90300720e+02, 2.22300720e+02],
  [1.50376404e+02, 1.93294495e+02, 2.25294495e+02],
  [1.50953125e+02, 1.93953125e+02, 2.25953125e+02]],

 ...,

 [[2.47031250e+01, 2.47031250e+01, 2.47031250e+01],
  [2.63285522e+01, 2.63285522e+01, 2.63285522e+01],
  [3.37168579e+01, 3.37168579e+01, 3.37168579e+01],
  ...,
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01]],
```

```
 [[3.64218750e+01, 3.64218750e+01, 3.64218750e+01],
  [3.72080688e+01, 3.72080688e+01, 3.72080688e+01],
  [4.07816772e+01, 4.07816772e+01, 4.07816772e+01],
  ...,
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01]],

 [[3.90000000e+01, 3.90000000e+01, 3.90000000e+01],
  [3.96015625e+01, 3.96015625e+01, 3.96015625e+01],
  [4.23359375e+01, 4.23359375e+01, 4.23359375e+01],
  ...,
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01],
  [3.30000000e+01, 3.30000000e+01, 3.30000000e+01]]],


...,


[[[2.60000000e+01, 1.20000000e+01, 1.20000000e+01],
  [2.44570312e+01, 1.04570312e+01, 1.04570312e+01],
  [2.09042969e+01, 8.14257812e+00, 8.14257812e+00],
  ...,
  [5.30000000e+01, 2.50000000e+01, 1.40000000e+01],
  [5.41425781e+01, 2.61425781e+01, 1.51425781e+01],
  [5.80000000e+01, 3.00000000e+01, 1.90000000e+01]],

 [[2.85781250e+01, 1.45781250e+01, 1.45781250e+01],
  [2.70351562e+01, 1.30351562e+01, 1.30351562e+01],
  [2.45465698e+01, 1.07207031e+01, 1.07207031e+01],
  ...,
  [5.38593750e+01, 2.58593750e+01, 1.48593750e+01],
  [5.50019531e+01, 2.70019531e+01, 1.60019531e+01],
  [5.88593750e+01, 3.08593750e+01, 1.98593750e+01]],

 [[3.35937500e+01, 1.80625000e+01, 1.80625000e+01],
  [3.14601135e+01, 1.71101990e+01, 1.65195312e+01],
  [2.89707031e+01, 1.49707031e+01, 1.42050781e+01],
  ...,
  [5.96509705e+01, 3.16509705e+01, 2.06509705e+01],
  [6.00428772e+01, 3.20428772e+01, 2.10428772e+01],
  [5.97656250e+01, 3.17656250e+01, 2.07656250e+01]],

 ...,

 [[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  ...,
  [3.92166138e+00, 3.92166138e+00, 3.92166138e+00],
  [7.94387817e+00, 7.94387817e+00, 7.94387817e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]],

 [[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  ...,
  [2.96667480e+00, 2.96667480e+00, 2.96667480e+00],
  [5.30395508e+00, 5.30395508e+00, 5.30395508e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]],

 [[0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  ...,
  [2.47656250e+00, 2.47656250e+00, 2.47656250e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
  [0.00000000e+00, 0.00000000e+00, 0.00000000e+00]]],


[[[1.20000000e+01, 4.80000000e+01, 7.40000000e+01],
  [1.20000000e+01, 4.80000000e+01, 7.40000000e+01],
  [1.20000000e+01, 4.80000000e+01, 7.40000000e+01],
  ...,
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02]],

 [[1.20000000e+01, 4.80000000e+01, 7.40000000e+01],
  [1.20073853e+01, 4.80073853e+01, 7.40073853e+01],
  [1.20409546e+01, 4.80409546e+01, 7.40409546e+01],
  ...,
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
  [4.30000000e+01, 9.70000000e+01, 1.61000000e+02]],

 [[1.20000000e+01, 4.80000000e+01, 7.40000000e+01],
```

```
        [1.20409546e+01, 4.80409546e+01, 7.40409546e+01],
        [1.22271118e+01, 4.82271118e+01, 7.42271118e+01],
        ...,
        [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
        [4.30000000e+01, 9.70000000e+01, 1.61000000e+02],
        [4.30000000e+01, 9.70000000e+01, 1.61000000e+02]],

       ...,

       [[1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        ...,
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01]],

       [[1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        ...,
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01]],

       [[1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        [1.50000000e+01, 1.10000000e+01, 1.20000000e+01],
        ...,
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01],
        [1.00000000e+00, 3.20000000e+01, 6.30000000e+01]]],


      [[[3.48144531e+01, 4.38144531e+01, 2.68144531e+01],
        [2.96660156e+01, 3.85546875e+01, 2.17773438e+01],
        [3.07109375e+01, 3.87109375e+01, 2.37109375e+01],
        ...,
        [6.67011719e+01, 1.07886719e+02, 1.54628906e+02],
        [6.62226562e+01, 1.07222656e+02, 1.51445312e+02],
        [6.31113281e+01, 1.04111328e+02, 1.48111328e+02]],

       [[3.70610428e+01, 4.60610428e+01, 2.90610428e+01],
        [3.08105469e+01, 3.96992188e+01, 2.29218750e+01],
        [3.11121941e+01, 3.91121941e+01, 2.41121941e+01],
        ...,
        [6.80167084e+01, 1.07379280e+02, 1.53336838e+02],
        [6.67949219e+01, 1.06586685e+02, 1.50237076e+02],
        [6.36835938e+01, 1.03539062e+02, 1.46966797e+02]],

       [[3.17436256e+01, 4.01694069e+01, 2.34565163e+01],
        [2.98805542e+01, 3.81950073e+01, 2.17047729e+01],
        [3.36024399e+01, 4.10282211e+01, 2.63153305e+01],
        ...,
        [7.30402832e+01, 1.09604736e+02, 1.53827393e+02],
        [7.15292969e+01, 1.08982422e+02, 1.50801498e+02],
        [6.81415100e+01, 1.05705963e+02, 1.47270416e+02]],

       ...,

       [[9.12762604e+01, 7.25898209e+01, 6.46278229e+01],
        [6.79436340e+01, 4.99436302e+01, 3.93694115e+01],
        [6.78729095e+01, 4.98729057e+01, 3.90341377e+01],
        ...,
        [4.94336205e+01, 3.61768837e+01, 3.98136024e+01],
        [5.04206352e+01, 3.96391983e+01, 4.63593750e+01],
        [5.52606544e+01, 4.38570709e+01, 5.11237602e+01]],

       [[7.37857132e+01, 5.56624184e+01, 4.85828362e+01],
        [6.47256317e+01, 4.71533661e+01, 3.80088348e+01],
        [7.11834717e+01, 5.36112099e+01, 4.40955849e+01],
        ...,
        [5.40664749e+01, 4.35013084e+01, 5.05535164e+01],
        [4.90478020e+01, 4.13600235e+01, 5.21647110e+01],
        [4.67969360e+01, 3.93633423e+01, 5.05020142e+01]],

       [[5.70000000e+01, 4.00000000e+01, 3.29628906e+01],
        [5.81132812e+01, 4.11132812e+01, 3.31132812e+01],
        [6.45878906e+01, 4.75878906e+01, 3.92167969e+01],
        ...,
        [6.08867188e+01, 5.21445312e+01, 6.18457031e+01],
        [4.95585938e+01, 4.51132812e+01, 5.87792969e+01],
        [3.74082031e+01, 3.34082031e+01, 4.74082031e+01]]]],
     dtype=float32),
array([3, 1, 1, 3, 4, 2, 2, 4, 4, 1, 2, 2, 0, 3, 1, 0, 2, 4, 2, 4, 0, 0,
       4, 1, 2, 3, 3, 4, 0, 2, 2, 0]))
```

In [ ]:
```
batch[0].shape # 32 images, 256x256 size, 3 channels(rgb)
```

Out[ ]:
```
(32, 256, 256, 3)
```

In [ ]:
```
# first image
batch[0][0]
```

Out[ ]:
```
array([[[0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        ...,
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ]],

       [[0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        ...,
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ]],

       [[0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        ...,
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ],
        [0.      , 0.      , 0.      ]],

       ...,

       [[0.      , 0.015625, 0.      ],
        [0.      , 0.015625, 0.      ],
        [0.      , 0.015625, 0.      ],
        ...,
        [0.      , 0.015625, 0.      ],
        [0.      , 0.015625, 0.      ],
        [0.      , 0.015625, 0.      ]],

       [[3.390625, 0.      , 0.      ],
        [3.390625, 0.      , 0.      ],
        [3.390625, 0.      , 0.      ],
        ...,
        [3.390625, 0.      , 0.      ],
        [3.390625, 0.      , 0.      ],
        [3.390625, 0.      , 0.      ]],

       [[5.59375 , 0.      , 0.      ],
        [5.59375 , 0.      , 0.      ],
        [5.59375 , 0.      , 0.      ],
        ...,
        [5.59375 , 0.      , 0.      ],
        [5.59375 , 0.      , 0.      ],
        [5.59375 , 0.      , 0.      ]]], dtype=float32)
```

In [ ]:
```
# min value in the array
batch[0][0].min()
```

Out[ ]:
```
0.0
```

In [ ]:
```
# max value in the array
batch[0][0].max()
```

Out[ ]:
```
249.68848
```

Data Preprocesssing

In [ ]:
```
data
```

Out[ ]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(
shape=(None,), dtype=tf.int32, name=None))>
```

In [ ]:
```
data = data.map(lambda x,y : (x/255,y))
```

In [ ]:
```
data.as_numpy_iterator().next()[0].min() # now the min value is 0
```

Out[ ]:
```
0.0
```

In [ ]:
```
data.as_numpy_iterator().next()[0].max() # now the max value is 1
```

Out[ ]:
```
1.0
```

In [ ]:
```
 len(data) # split by 11b+2+2
```

Out[ ]:
```
38
```

In [ ]:
```
# spliting test, train data

train = data.take(28)
val = data.skip(28).take(5)
test = data.skip(33).take(5)
```

Model Training

Building Model

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

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

In [ ]:
```
# 1st layer
"""
convo network with 16 filters of size 3*3 with stride 1
activation fn - relu
input shape- 256*256*3 . here 3 is 3 channel(rgb)

"""
model.add(Conv2D(16, (3, 3), 1, activation="relu", input_shape=(256, 256, 3)))
model.add(
    MaxPooling2D()
)  # max pooling layer of size (2,2). takes the maximum over the input window
model.add(Dropout(0.25))  # Add dropout after the first convolutional layer

# 2nd layer
"""
convo network with 32 filters of size 3*3 with stride 1
activation fn - relu

"""
model.add(Conv2D(32, (3, 3), 1, activation="relu"))
model.add(
    MaxPooling2D()
)  # max pooling layer of size (2,2). takes the maximum over the input window
model.add(Dropout(0.25))  # Add dropout after the second convolutional layer

# 3rd layer
"""
convo network with 16 filters of size 3*3 with stride 1
activation fn - relu

"""

model.add(Conv2D(16, (3, 3), 1, activation="relu"))
model.add(
    MaxPooling2D()
)  # max pooling layer of size (2,2). takes the maximum over the input window
model.add(Dropout(0.25))  # Add dropout after the third convolutional layer

# flattening to 1D array
model.add(Flatten())

# building dense(fully connected) neuron network with 256 neurons
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))  # Add dropout after the dense layer

# output layer with 4 neurons (4-class classification - so softmax).
model.add(Dense(5, activation="softmax"))
```

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

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2  (None, 127, 127, 16)      0
 D)

 dropout (Dropout)           (None, 127, 127, 16)      0

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)

 dropout_1 (Dropout)         (None, 62, 62, 32)        0

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 16)        0
 g2D)

 dropout_2 (Dropout)         (None, 30, 30, 16)        0

 flatten (Flatten)           (None, 14400)             0

 dense (Dense)               (None, 256)               3686656

 dropout_3 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 5)                 1285

=================================================================
Total params: 3697653 (14.11 MB)
Trainable params: 3697653 (14.11 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training Model

In [ ]:
```
logdir = 'logs'
```

In [ ]:
```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

In [ ]:
```
hist = model.fit(
    train, epochs=40, validation_data=val, callbacks=[tensorboard_callback])
```

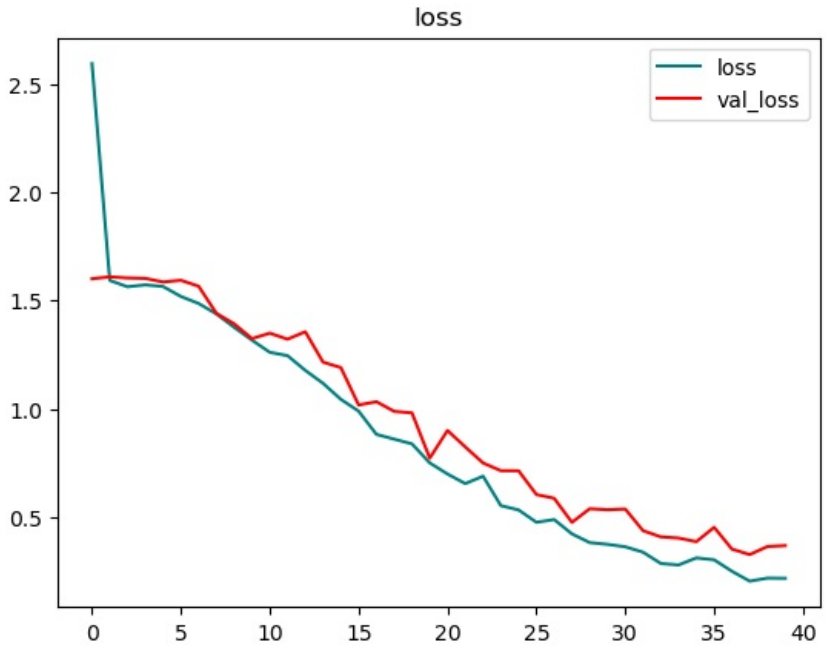
```
Epoch 1/40
28/28 [==============================] - 67s 2s/step - loss: 1.9634 - accuracy: 0.2455 - val_loss: 1.6073 - val
_accuracy: 0.2562
Epoch 2/40
28/28 [==============================] - 61s 2s/step - loss: 1.5503 - accuracy: 0.2857 - val_loss: 1.5943 - val
_accuracy: 0.2625
Epoch 3/40
28/28 [==============================] - 40s 1s/step - loss: 1.5215 - accuracy: 0.3292 - val_loss: 1.5443 - val
_accuracy: 0.4125
Epoch 4/40
28/28 [==============================] - 41s 1s/step - loss: 1.4739 - accuracy: 0.3806 - val_loss: 1.5057 - val
_accuracy: 0.4938
Epoch 5/40
28/28 [==============================] - 35s 1s/step - loss: 1.4103 - accuracy: 0.4431 - val_loss: 1.4262 - val
_accuracy: 0.4688
Epoch 6/40
28/28 [==============================] - 35s 1s/step - loss: 1.3063 - accuracy: 0.4944 - val_loss: 1.3323 - val
_accuracy: 0.5437
Epoch 7/40
28/28 [==============================] - 49s 2s/step - loss: 1.1995 - accuracy: 0.5301 - val_loss: 1.2266 - val
_accuracy: 0.5562
Epoch 8/40
28/28 [==============================] - 43s 1s/step - loss: 1.0853 - accuracy: 0.5636 - val_loss: 1.0951 - val
_accuracy: 0.5938
Epoch 9/40
28/28 [==============================] - 45s 1s/step - loss: 0.9898 - accuracy: 0.6306 - val_loss: 0.9127 - val
_accuracy: 0.7188
Epoch 10/40
28/28 [==============================] - 44s 2s/step - loss: 0.8802 - accuracy: 0.6629 - val_loss: 0.9952 - val
_accuracy: 0.6687
Epoch 11/40
28/28 [==============================] - 48s 2s/step - loss: 0.8127 - accuracy: 0.6942 - val_loss: 0.9897 - val
_accuracy: 0.6500
Epoch 12/40
28/28 [==============================] - 51s 2s/step - loss: 0.7774 - accuracy: 0.7176 - val_loss: 0.8579 - val
_accuracy: 0.6875
Epoch 13/40
28/28 [==============================] - 51s 2s/step - loss: 0.6417 - accuracy: 0.7556 - val_loss: 0.7912 - val
_accuracy: 0.7250
Epoch 14/40
28/28 [==============================] - 52s 2s/step - loss: 0.5469 - accuracy: 0.7958 - val_loss: 0.6187 - val
_accuracy: 0.8062
Epoch 15/40
28/28 [==============================] - 52s 2s/step - loss: 0.5153 - accuracy: 0.8080 - val_loss: 0.6016 - val
_accuracy: 0.7937
Epoch 16/40
28/28 [==============================] - 53s 2s/step - loss: 0.4323 - accuracy: 0.8516 - val_loss: 0.5970 - val
_accuracy: 0.8125
Epoch 17/40
28/28 [==============================] - 52s 2s/step - loss: 0.3844 - accuracy: 0.8650 - val_loss: 0.5217 - val
_accuracy: 0.8062
Epoch 18/40
28/28 [==============================] - 51s 2s/step - loss: 0.3967 - accuracy: 0.8739 - val_loss: 0.3933 - val
_accuracy: 0.8687
Epoch 19/40
28/28 [==============================] - 51s 2s/step - loss: 0.3620 - accuracy: 0.8705 - val_loss: 0.5126 - val
_accuracy: 0.8313
Epoch 20/40
28/28 [==============================] - 51s 2s/step - loss: 0.3520 - accuracy: 0.8739 - val_loss: 0.4980 - val
_accuracy: 0.8313
Epoch 21/40
28/28 [==============================] - 52s 2s/step - loss: 0.2713 - accuracy: 0.9062 - val_loss: 0.3270 - val
_accuracy: 0.8750
Epoch 22/40
28/28 [==============================] - 52s 2s/step - loss: 0.2444 - accuracy: 0.9196 - val_loss: 0.3705 - val
_accuracy: 0.8938
Epoch 23/40
28/28 [==============================] - 54s 2s/step - loss: 0.2158 - accuracy: 0.9375 - val_loss: 0.2986 - val
_accuracy: 0.9062
Epoch 24/40
28/28 [==============================] - 52s 2s/step - loss: 0.1885 - accuracy: 0.9386 - val_loss: 0.3686 - val
_accuracy: 0.8875
Epoch 25/40
28/28 [==============================] - 66s 2s/step - loss: 0.2291 - accuracy: 0.9208 - val_loss: 0.3597 - val
_accuracy: 0.9062
Epoch 26/40
28/28 [==============================] - 62s 2s/step - loss: 0.1891 - accuracy: 0.9364 - val_loss: 0.3924 - val
_accuracy: 0.8750
Epoch 27/40
28/28 [==============================] - 63s 2s/step - loss: 0.1506 - accuracy: 0.9598 - val_loss: 0.3549 - val
_accuracy: 0.9000
Epoch 28/40
28/28 [==============================] - 839s 31s/step - loss: 0.1476 - accuracy: 0.9565 - val_loss: 0.3384 - v
al_accuracy: 0.9250
Epoch 29/40
28/28 [==============================] - ETA: 0s - loss: 0.1307 - accuracy: 0.9576
```

In [ ]:
```
hist.history.keys()
```

Out[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

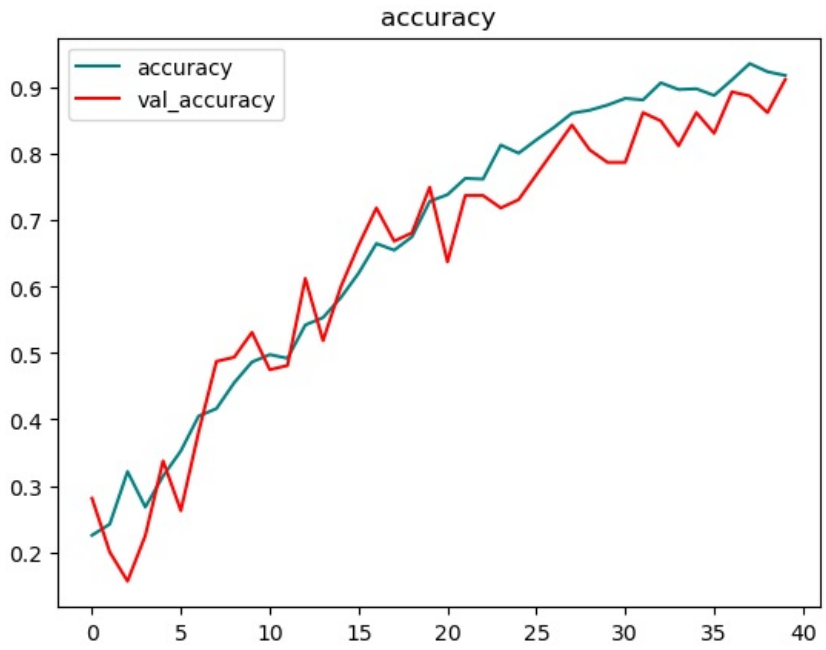
In [ ]:
```
import matplotlib.pyplot as plt
plt.plot(hist.history["loss"], color="teal", label="loss")
plt.plot(hist.history["val_loss"], color="red", label="val_loss")
plt.legend()
plt.title("loss")
```

Out[ ]:
```
Text(0.5, 1.0, 'loss')
```



In [ ]:
```
plt.plot(hist.history["accuracy"], color="teal", label="accuracy")
plt.plot(hist.history["val_accuracy"], color="red", label="val_accuracy")
plt.legend()
plt.title("accuracy")
```

Out[ ]:
```
Text(0.5, 1.0, 'accuracy')
```



In [ ]:
```
model.save(os.path.join("models", "debris_classifier_model_4.h5"))
```

```
c:\Users\mahig\anaconda3\Lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

Testing

In [ ]:
```
from tensorflow.keras.metrics import Precision, Recall,  CategoricalAccuracy
```

In [ ]:
```
pre = Precision()
```

```
re = Recall()
acc = CategoricalAccuracy()
```

In [ ]:
```
for batch in test.as_numpy_iterator():

    x, y = batch
    yhat = model.predict(x)
    # Assuming y contains the integer class labels (0 to 4) for 5 classes
    num_classes = 5
    y_one_hot = tf.one_hot(y, num_classes)

    # Now, you can use y_one_hot in the metrics update_state calls
    pre.update_state(y_one_hot, yhat)
    re.update_state(y_one_hot, yhat)
    acc.update_state(y_one_hot, yhat)
    # pre.update_state(y, yhat)
    # re.update_state(y, yhat)
    # acc.update_state(y, yhat)
```

```
1/1 [==============================] - 1s 558ms/step
1/1 [==============================] - 0s 222ms/step
1/1 [==============================] - 0s 222ms/step
1/1 [==============================] - 0s 229ms/step
1/1 [==============================] - 0s 143ms/step
```

In [ ]:
```
print(
    f"""
precision = {pre.result().numpy()},
Recall = {re.result().numpy()}
Accuracy = {acc.result().numpy()}
"""
)
```

```
precision = 0.9032257795333862,
Recall = 0.8175182342529297
Accuracy = 0.8613138794898987
```

Importing the saved model

In [ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
saved_model = tf.keras.models.load_model(r'C:\Users\mahig\OneDrive\Desktop\Project\models\debris_classifier_mod
```

INPUT

In [ ]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def plot_image(file_path):
    # Load the image
    img = mpimg.imread(file_path)

    # Display the image
    plt.imshow(img)
    plt.axis('off')  # Turn off axis labels
    plt.show()

if __name__ == "__main__":
    # Replace 'your_image_path.jpg' with the actual file path of your image
    image_path = r'C:\Users\mahig\OneDrive\Desktop\Predicated Output\WhatsApp Image 2023-11-16 at 11.58.02_fbe1
    plot_image(image_path)
```

In [ ]:

```python
import os
import numpy as np
from PIL import Image

img_path = r"C:\Users\mahig\OneDrive\Desktop\Predicated Output\WhatsApp Image 2023-11-16 at 11.58.02_fbe15a8c.j

# Check if the file exists
if os.path.exists(img_path):
    try:
        img = Image.open(img_path)
        if img:
            # Resize the image to match the expected input shape of the model
            img = img.resize((256, 256))

            img_array = np.array(img)  # Convert the image to a numpy array

            # Expand dimensions to match the expected input shape
            img_array = np.expand_dims(img_array, axis=0)

            # Normalize the image
            img_array = img_array / 255.0

            # Make the prediction
            prediction = model.predict(img_array)

            # Get the predicted class label
            predicted_class = np.argmax(prediction, axis=1)

            # Print the predicted class
            print("Predicted Class:", predicted_class)

            # Print additional information based on predicted class
            if predicted_class == 0:
                print("Predicted Output: Bottle")
            elif predicted_class == 1:
                print("Predicted Output: Fishnet")
                # Add more conditions for other classes as needed
            elif predicted_class == 2:
                print("Predicted Output: Metal")
                # Add more conditions for other classes as needed
            elif predicted_class == 3:
                print("Predicted Output: Plastic")
                # Add more conditions for other classes as needed
            elif predicted_class == 4:
                print("Predicted Output: Tyre")
                # Add more conditions for other classes as needed

        else:
            print(f"Unable to open the image file at {img_path}.")
    except Exception as e:
        print(f"Error: {e}")
else:
    print(f"No file found at {img_path}. Please provide the correct path to the image.")
```

```
1/1 [==============================] - 0s 63ms/step
Predicted Class: [3]
Predicted Output: Plastic
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js