```
SUSMITHA M       21BKT0025
M RAJKUMAR       21BCT0264
G RITHWIK REDDY  21BCT0241
K.K.S.SRAVANTHI  21BCT0249
```

```
from tensorflow.keras.applications import EfficientNetB7
model = EfficientNetB7(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb7.h5
268326632/268326632 [==============================] - 3s 0us/step
```

```
model.summary()
```

```
block7d_expand_bn (BatchNormal  (None, 19, 19, 3840  15360      ['block7d_expand_conv[0][0]']
ization)                        )

block7d_expand_activation (Act  (None, 19, 19, 3840  0          ['block7d_expand_bn[0][0]']
ivation)                        )

block7d_dwconv (DepthwiseConv2  (None, 19, 19, 3840  34560      ['block7d_expand_activation[0][0]
D)                              )                                ']

block7d_bn (BatchNormalization  (None, 19, 19, 3840  15360      ['block7d_dwconv[0][0]']
)                               )

block7d_activation (Activation  (None, 19, 19, 3840  0          ['block7d_bn[0][0]']
)                               )

block7d_se_squeeze (GlobalAver  (None, 3840)        0           ['block7d_activation[0][0]']
agePooling2D)

block7d_se_reshape (Reshape)    (None, 1, 1, 3840)   0          ['block7d_se_squeeze[0][0]']

block7d_se_reduce (Conv2D)      (None, 1, 1, 160)    614560      ['block7d_se_reshape[0][0]']

block7d_se_expand (Conv2D)      (None, 1, 1, 3840)   618240      ['block7d_se_reduce[0][0]']

block7d_se_excite (Multiply)    (None, 19, 19, 3840  0          ['block7d_activation[0][0]',
                                )                                'block7d_se_expand[0][0]']

block7d_project_conv (Conv2D)   (None, 19, 19, 640)  2457600     ['block7d_se_excite[0][0]']

block7d_project_bn (BatchNorma  (None, 19, 19, 640)  2560       ['block7d_project_conv[0][0]']
lization)

block7d_drop (Dropout)          (None, 19, 19, 640)  0          ['block7d_project_bn[0][0]']

block7d_add (Add)               (None, 19, 19, 640)  0          ['block7d_drop[0][0]',
                                                                 'block7c_add[0][0]']

top_conv (Conv2D)               (None, 19, 19, 2560  1638400     ['block7d_add[0][0]']
                                )

top_bn (BatchNormalization)     (None, 19, 19, 2560  10240       ['top_conv[0][0]']
                                )

top_activation (Activation)     (None, 19, 19, 2560  0          ['top_bn[0][0]']
                                )

avg_pool (GlobalAveragePooling  (None, 2560)        0           ['top_activation[0][0]']
2D)

top_dropout (Dropout)           (None, 2560)         0          ['avg_pool[0][0]']

predictions (Dense)             (None, 1000)         2561000     ['top_dropout[0][0]']

==================================================================================================
Total params: 66,658,687
Trainable params: 66,347,960
Non-trainable params: 310,727
__________________________________________________________________________________________________
```

```
from google.colab import files
import cv2
import matplotlib.pyplot as plt
```

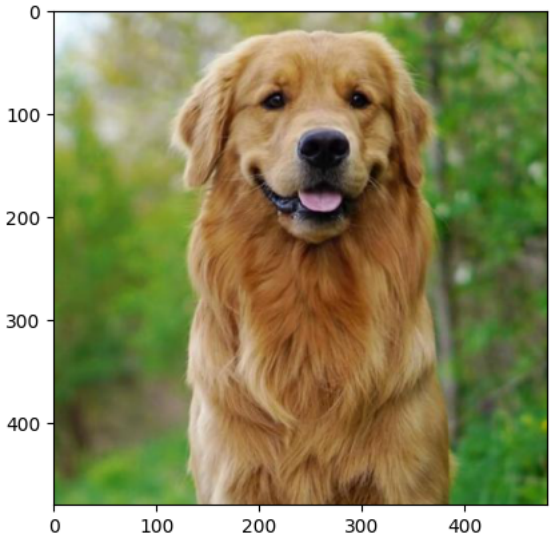
```
uploaded= files.upload()
```

```
Choose Files golden-retriever-dog.jpg
```
- **golden-retriever-dog.jpg**(image/jpeg) - 27327 bytes, last modified: 3/29/2023 - 100% done

```
Saving golden-retriever-dog.jpg to golden-retriever-dog.jpg
```

```
import cv2
import numpy as np
from matplotlib.pyplot import imread
from matplotlib.pyplot import imshow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.imagenet_utils import decode_predictions
from tensorflow.keras.applications.imagenet_utils import preprocess_input


img_path = 'golden-retriever-dog.jpg'
img = cv2.imread(img_path)
img = cv2.resize(img, (600,600))

x = np.expand_dims(img, axis=0)
x = preprocess_input(x)

print('Input image shape:', x.shape)

my_image = imread(img_path)
imshow(my_image)
```

```
Input image shape: (1, 600, 600, 3)
<matplotlib.image.AxesImage at 0x7f99d86bf1c0>
```




```
preds=model.predict(x)
print("predicted class: ", preds )
```

```
1/1 [==============================] - 11s 11s/step
predicted class:  [[1.90080274e-04 1.91119281e-04 1.26521540e-04 1.42403616e-04
  1.37726587e-04 2.77265528e-04 1.73037552e-04 1.22635902e-04
  2.76932551e-04 1.96920708e-04 2.31213329e-04 2.46320909e-04
  2.84947833e-04 2.25320706e-04 1.89596351e-04 1.76066329e-04
  8.29049313e-05 2.20031070e-04 1.39411350e-04 4.32808913e-04
  1.46168459e-04 1.58234921e-04 3.07301088e-04 2.07294055e-04
  2.27515411e-04 1.38055490e-04 1.11196663e-04 1.47478771e-04
  2.86685477e-04 1.58703551e-04 2.01753835e-04 2.46926298e-04
  1.07615379e-04 1.89549537e-04 4.26147366e-04 1.78010974e-04
  1.91868938e-04 1.03216516e-04 1.90995386e-04 1.42427249e-04
  1.56659517e-04 2.13005536e-04 2.33719576e-04 1.66519079e-04
  3.10547999e-04 1.82245931e-04 1.38237694e-04 1.96049936e-04
  1.76323243e-04 9.69701578e-05 9.59779863e-05 1.43784229e-04
  1.96378533e-04 3.14314559e-04 1.99254486e-04 2.16131783e-04
  1.25716542e-04 1.96648223e-04 1.96000023e-04 3.32268042e-04
  3.83602339e-04 1.48617182e-04 2.33708895e-04 2.60309433e-04
  4.42192337e-04 2.21076232e-04 2.46671640e-04 2.43968316e-04
  2.45936651e-04 1.57643575e-04 2.08430924e-04 1.75258858e-04
  1.10314584e-04 1.41198165e-04 1.00116849e-04 1.25122679e-04
  1.62756172e-04 1.22260812e-04 7.85656812e-05 1.94732900e-04
  1.18872660e-04 1.37783485e-04 1.63303601e-04 1.80570394e-04
  1.51031156e-04 2.18280038e-04 1.91286847e-04 1.21245415e-04
  3.22768406e-04 2.12384024e-04 1.71169115e-04 1.22797283e-04
```

```
3.60740640e-04 4.16021066e-04 1.07844178e-04 2.23285359e-04
1.92822437e-04 2.17002322e-04 8.36538748e-05 1.46613398e-04
2.62080983e-04 1.65118632e-04 1.42154895e-04 2.33555154e-04
1.35156268e-04 1.19580116e-04 1.51503322e-04 1.29273380e-04
1.97910253e-04 1.23608115e-04 2.31451791e-04 2.59969442e-04
1.54158974e-04 1.74027373e-04 3.11776384e-04 2.21180628e-04
1.06565494e-04 1.66796424e-04 1.51621251e-04 1.52121080e-04
7.94296502e-05 1.78193572e-04 1.98791749e-04 1.79445109e-04
7.76446614e-05 1.22955244e-04 1.28188985e-04 2.02803247e-04
1.91148254e-04 1.69618550e-04 1.97390764e-04 2.51721358e-04
3.07963695e-04 1.50789085e-04 1.41822195e-04 2.02016454e-04
1.22931800e-04 2.25946234e-04 1.30582470e-04 1.59463802e-04
2.65621085e-04 2.38790904e-04 2.41424656e-04 2.13803316e-04
1.66074547e-04 3.43741500e-04 2.31096274e-04 1.24007332e-04
1.72127227e-04 2.14526328e-04 1.52385066e-04 2.63729016e-04
1.91811327e-04 1.09759152e-04 2.80909211e-04 6.59350771e-05
2.99651321e-04 2.11639344e-04 3.08532530e-04 5.77485887e-04
7.67548554e-05 6.43618259e-05 1.45005790e-04 9.17848083e-05
1.37371506e-04 1.20651159e-04 5.34729101e-04 8.14418672e-05
2.70057964e-04 1.55635295e-04 4.85099154e-04 1.33189722e-04
6.92832109e-05 1.39170515e-04 2.06612545e-04 2.87214178e-04
2.02411000e-04 1.56918642e-04 4.43222059e-04 5.49694851e-05
9.06925634e-05 2.59728811e-04 1.72580054e-04 1.01414727e-04
8.86773458e-04 1.44474162e-03 1.02748000e-03 2.19457492e-04
2.02106443e-04 1.98079622e-04 2.94492667e-04 1.03655492e-03
1.74193250e-04 3.30115872e-04 1.28273125e-04 7.04879712e-05
5.04815020e-04 2.06341414e-04 2.80386070e-04 6.89993394e-05
3.22070438e-04 1.69633422e-04 1.60616270e-04 1.88127349e-04
1.52387671e-04 6.77272072e-03 9.82302474e-04 7.63491750e-01
1.47396594e-03 3.88464803e-04 1.03684848e-04 2.01253803e-04
1.16491210e-04 4.78317495e-03 5.66283124e-04 7.73360895e-04
4.93985484e-04 9.69273460e-05 3.91243404e-04 6.53935887e-04
1.12444512e-03 7.05215280e-05 3.84389865e-03 1.43759680e-04
3.97556374e-04 5.33918828e-05 1.70857762e-04 1.01922604e-04
```

```
decode_predictions(preds,top=3)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
[[('n02099601', 'golden_retriever', 0.76349175),
  ('n04409515', 'tennis_ball', 0.010787531),
  ('n02099267', 'flat-coated_retriever', 0.0067727207)]]
```

So the final result is Golden Retriever has been predicted with 76.35% accuracy