```
import os
import pandas as pd
import numpy as np

from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPool
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Flatten, Dr
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras import regularizers
from tensorflow.keras.models import Model
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
import tensorflow as tf
import logging
import warnings
import keras

from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, GlobalA

logging.getLogger("tensorflow").setLevel(logging.ERROR)

```

```
--NORMAL--
```

```

```

```

print(os.listdir())
train_path='/content/drive/MyDrive/Fillterd/train/'
train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applicatior
    .flow_from_directory(directory=train_path, target_size=(224,224), classes=

test_path='/content/drive/MyDrive/Fillterd/test/'
test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications
   .flow_from_directory(directory=test_path, target_size=(224,224), classes=[

```

```

warnings.simplefilter("ignore")
model_name='EfficientNetB3'
base_model=tf.keras.applications.efficientnet.EfficientNetB3(include_top=False,
# Note you are always told NOT to make the base model trainable initially- that
base_model.trainable=True
x=base_model.output
#
# x = GlobalAveragePooling2D()(x)
x = Flatten()(x)
# loss='binary_crossentropy' activation='sigmoid'
x=BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(1024, kernel_regularizer = regularizers.l2(l = 0.016),activity_regula
               bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
```

```
x=Dropout(rate=.3, seed=123)(x)
x = Dense(128, kernel_regularizer = regularizers.l2(l = 0.016),activity_regular
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.25, seed=123)(x)
output=Dense(2, activation='sigmoid')(x)
model=Model(inputs=base_model.input, outputs=output)
lr=.001 # start with this learning rate
model.compile(Adamax(learning_rate=lr), loss='binary_crossentropy', metrics=['a
# model.compile(optimizer='adam',loss = keras.losses.SparseCategoricalCrossentr

# rlronp=tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss", factor=0.5, pa
# estop=tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=4, verbos
mc = ModelCheckpoint('efNetB32.h5', monitor='val_accuracy', mode='max', verbose=

callbacks=[mc]

epochs=10
history2=model.fit(x=train_batches,  epochs=epochs, callbacks=callbacks,  valid
               validation_steps=None,  verbose=1, shuffle=True,  initial_epoch=0

```

```

# test_path='/content/drive/MyDrive/Fillterd/test/'
# test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications
#    .flow_from_directory(directory=test_path, target_size=(224,224), classes=[


from keras.models import load_model

saved_model = load_model('/content/drive/MyDrive/ComputerVision/efNetB3.h5')

test_loss, test_acc = saved_model.evaluate(test_batches, verbose=1)
print('Test: %.3f' % (test_acc))
print('Loss: %.3f' % (test_loss))
```

```
# To Plot
import pickle
with open('/content/drive/MyDrive/ComputerVision/trainHistoryDict', 'wb') as fi
        pickle.dump(history2.history, file_pi)

from keras.models import load_model

saved_model = load_model('/content/drive/MyDrive/ComputerVision/efNetB3.h5')

h=np.load('/content/drive/MyDrive/ComputerVision/trainHistoryDict',allow_pickle=
# h
```

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])
```

```
# Add traces
fig.add_trace(
    go.Scatter( y=h['val_loss'], name="test loss"),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter( y=h['loss'], name="train loss"),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter( y=h['val_accuracy'], name="test accuracy"),
    secondary_y=True,
)

fig.add_trace(
    go.Scatter( y=h['accuracy'], name="train accuracy"),
    secondary_y=True,
)

# Add figure title
fig.update_layout(
    title_text="Loss/Accuracy of EfficientNetB3 Model"
)

# Set x-axis title
fig.update_xaxes(title_text="Epoch")

# Set y-axes titles
fig.update_yaxes(title_text="<b>Loss</b>", secondary_y=False)
fig.update_yaxes(title_text="<b>Accuracy</b>", secondary_y=True)

fig.show()
```

```
from matplotlib import pyplot as plt
import pandas as pd

pd.DataFrame(h).plot(figsize=(8,5))
plt.show()

# plt.plot(h['accuracy'])
# plt.plot(h['val_accuracy'])
# plt.title('model accuracy')
# plt.ylabel('accuracy')
# plt.xlabel('epoch')
# plt.legend(['train', 'val'], loc='upper right')
# plt.show()

# plt.plot(h['loss'])
# plt.plot(h['val_loss'])
# plt.title('model loss')
# plt.ylabel('loss')
```

```
# plt.xlabel('epoch')
# plt.legend(['train', 'val'], loc='upper right')
# plt.show()
```

Segmentation

```
from sklearn.cluster import KMeans
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import torch
import glob
import cv2
import os


def getColor(id):
  if id ==0:
    return [0,255,255]
  elif id ==1:
    return [139,131,120]
  elif id ==2:
    return [227,207,87]
  elif id ==3:
    return [0,0,139]
  elif id ==4:
    return [139,35,35]
  elif id ==5:
    return [152,245,255]
  elif id ==6:
    return [127,255,0]
  elif id ==7:
    return [205,91,69]
  elif id ==8:
    return [0,205,205]
  elif id ==9:
    return [154,50,205]
  elif id ==10:
    return [205,16,118]
  elif id ==11:
    return [0,201,87]

model =torch.hub.load('ultralytics/yolov5','custom',path='/content/drive/MyDrive

model.conf = 0.30  # confidence threshold (0-1)
model.iou = 0.30  # NMS IoU threshold (0-1)

os.chdir('/content/drive/MyDrive/ACV/Test/Images/')
images = glob.glob('*')


```

```
 1
 2 for img in images:
 3   e = model(img, size=640)
 4
 5   original_image = cv2.imread(img)
 6
 7   img2=cv2.cvtColor(original_image,cv2.COLOR_BGR2RGB)
 8
 9   HSVimg = cv2.cvtColor(img2, cv2.COLOR_RGB2HSV)
10   black = np.zeros(original_image.shape, dtype=np.uint8)
11
12   bla=[0,0,0]
13   for i in range(len(e.pandas().xyxy[0]['class'])):
14     color = getColor(class_id)
15     xmin = int(e.pandas().xyxy[0]['xmin'][i])
16     xmax = int(e.pandas().xyxy[0]['xmax'][i])
17     ymin = int(e.pandas().xyxy[0]['ymin'][i])
18     ymax = int(e.pandas().xyxy[0]['ymax'][i])
19     class_id = int(e.pandas().xyxy[0]['class'][i])
20     subimg = HSVimg[ymin:ymax,xmin:xmax]
21     vectorized = subimg.reshape((-1,3))
22     vectorized = np.float32(vectorized)
23     criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
24     K = 2
25     attempts=10
26     ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_
27     # convert back to 8 bit values
28     centers = np.uint8(center)
29     res = center[label.flatten()]
30     result_image = res.reshape((subimg.shape))
31     # flatten the labels array
32     labels = label.flatten()
33
34     # disable only the cluster number 2 (turn the pixel into black)
35     masked_image = np.copy(subimg)
36     # convert to the shape of a vector of pixel values
37     masked_image = masked_image.reshape((-1, 3))
38     # color (i.e cluster) to disable
39
40     if str(e.pandas().xyxy[0]['name'][i])=='Bicycle':
41       temp = color
42       color = bla
43       bla = temp
44       # print('swap is done')
45     df = pd.DataFrame(labels)
46     if df.value_counts()[0] > df.value_counts()[1]:
47       masked_image[labels == 0] = color
48       masked_image[labels == 1] = bla
49     else:
50       masked_image[labels == 1] = color
51       masked_image[labels == 0] = bla
52
53     bla=[0,0,0]
```

```
54
55    # convert back to original shape
56    masked_image = masked_image.reshape(subimg.shape)
57    # show the image
58
59    black[ymin:ymax,xmin:xmax] = masked_image
60    # cv2.imwrite('Severe%d.png'%(index),img)
61  cv2.imwrite('/content/drive/MyDrive/ACV/Seg/'+img,black)
62  # cv2_imshow(original_image)
63  # cv2_imshow(black)
```