```
import·numpy·as·np
import·pandas·as·pd·
import joblib
import matplotlib.pyplot as plt
import cv2 as cv
from path import Path
import os
import tensorflow as tf
import glob
import tensorflow_hub as hub
import os
import pydicom as dicom
from pydicom.pixel_data_handlers.util import apply_voi_lut
import tensorflow as tf
from tensorflow import keras
from keras import layers
from tqdm import tqdm
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.utils import to_categorical
from pydicom import dcmread
import nibabel as nib
import pickle
```

```
train_df = pd.read_csv("../input/rsna-2022-cervical-spine-fracture-detection/train.csv")
```

```
train_df
```

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
2014	1.2.826.0.1.3680043.21684	1	0	1	0	0	0	1	1
2015	1.2.826.0.1.3680043.4786	1	0	0	0	0	0	0	1
2016	1.2.826.0.1.3680043.14341	0	0	0	0	0	0	0	0
2017	1.2.826.0.1.3680043.12053	0	0	0	0	0	0	0	0
2018	1.2.826.0.1.3680043.18786	1	0	0	0	0	0	0	1

2019 rows × 9 columns

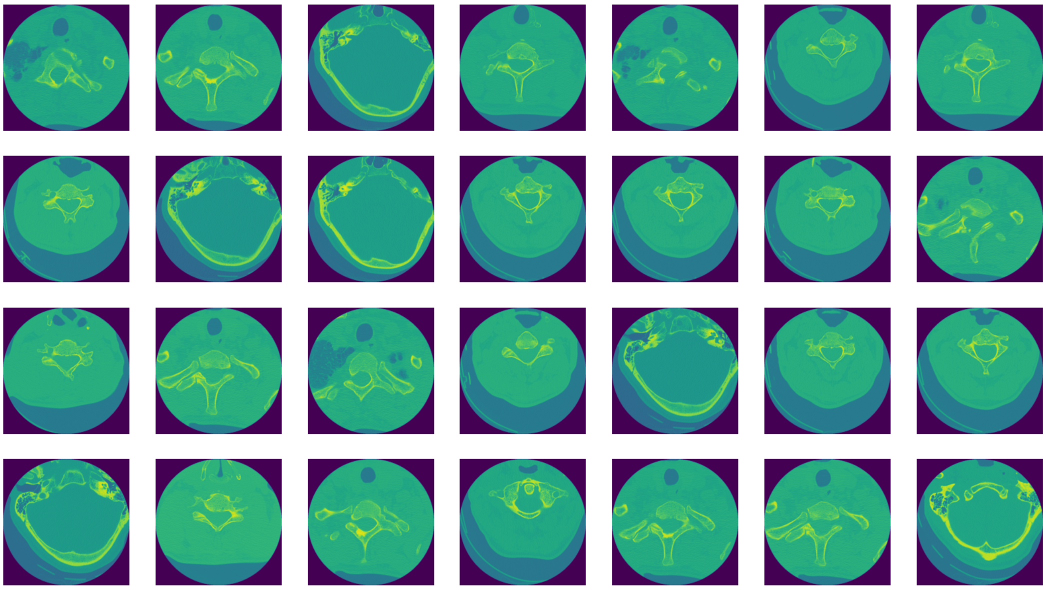
```
def load_dicom(path):
    img=dicom.dcmread(path)
    data=img.pixel_array
    data=data-np.min(data)
    if np.max(data) != 0:
        data=data/np.max(data)
    data=(data*255).astype(np.uint8)
    return data
```

```
def listdirs(folder):
    return [d for d in os.listdir(folder) if os.path.isdir(os.path.join(folder, d))]
```

```
train_dir='../input/rsna-2022-cervical-spine-fracture-detection/train_images'
patients = sorted(os.listdir(train_dir))
patients[:5]
```

```
['1.2.826.0.1.3680043.10001',
 '1.2.826.0.1.3680043.10005',
 '1.2.826.0.1.3680043.10014',
 '1.2.826.0.1.3680043.10016',
 '1.2.826.0.1.3680043.10032']
```

```
image_file = glob.glob("../input/rsna-2022-cervical-spine-fracture-detection/train_images/1.2.826.0.1.3680043.10001/*.dcm")
plt.figure(figsize=(20, 20))

for i in range(28):
    ax = plt.subplot(7, 7, i + 1)
    # specify your dcm image path
    image_path = image_file[i]

    image = load_dicom(image_path)

    plt.axis('off')

    plt.imshow(image)
```



```
from pydicom.data import get_testdata_files
trainset=[]
trainlabel=[]
trainidt=[]
limit = 2019
for i in tqdm(range(len(train_df))):
    idt=train_df.loc[i,'StudyInstanceUID']


    path=os.path.join(train_dir,idt)

    for im in os.listdir(path):


        dc = dicom.read_file(os.path.join(path,im))
        if dc.file_meta.TransferSyntaxUID.name =='JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14 [Selection Value 1]
            continue
        try:
            img=load_dicom(os.path.join(path,im))
        except:
            continue
#         ds = decode(os.path.join(path,im))
#         print(ds)
#         ds.decompress("pylibjpeg")

        img=np.resize(img,(64,64))
        image=img_to_array(img)
        image=image/255.0
        saved_csv_name = str(im) + ".csv"
        np.savetxt(saved_csv_name, image[:,:,0], delimiter = ",")
        image=img/255
        image = tf.expand_dims(image, axis=-1)
        image = tf.image.grayscale_to_rgb(image)
        trainset+=[image]
        cur_label=[]
        cur_label.append(train_df.loc[i,'C1'])
        cur_label.append(train_df.loc[i,'C2'])
        cur_label.append(train_df.loc[i,'C3'])
        cur_label.append(train_df.loc[i,'C4'])
        cur_label.append(train_df.loc[i,'C5'])
        cur_label.append(train_df.loc[i,'C6'])
        cur_label.append(train_df.loc[i,'C7'])
        trainlabel+=[cur_label]
        trainidt+=[idt]
    i+=1
    if i==limit +1:
        break
```

```
      100%|---------------| 2019/2019 [00:31<4:21:28,  78.79s/it]
```

```
trainset[0].shape
```

```
     TensorShape([64, 64, 3])
```

```
y=np.array(trainlabel)
Y_train=y
X_train=np.array(trainset)
```

```
test_df = ['1.2.826.0.1.3680043.22327', '1.2.826.0.1.3680043.25399', '1.2.826.0.1.3680043.5876']
```

```
test_dir='../input/rsna-2022-cervical-spine-fracture-detection/test_images'
testset=[]
testidt=[]
for i in tqdm(range(len(test_df))):
    idt=test_df[i]
    path=os.path.join(test_dir,idt)

    for im in os.listdir(path):
        dc = dicom.read_file(os.path.join(path,im))

        if dc.file_meta.TransferSyntaxUID.name =='JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14 [Selection Value 1]
            #print("hhh")
            continue
        img=load_dicom(os.path.join(path,im))

        img=cv.resize(img,(64,64))
        image=img_to_array(img)
        image=img/255.0
        image = tf.expand_dims(image, axis=-1)
        image = tf.image.grayscale_to_rgb(image)
        testset+=[image]
        testidt+=[idt]
```

```
     100%|██████████| 3/3 [00:19<00:00,  6.64s/it]
```

```
len(testset)
```

```
     771
```

```
X_train.shape
```

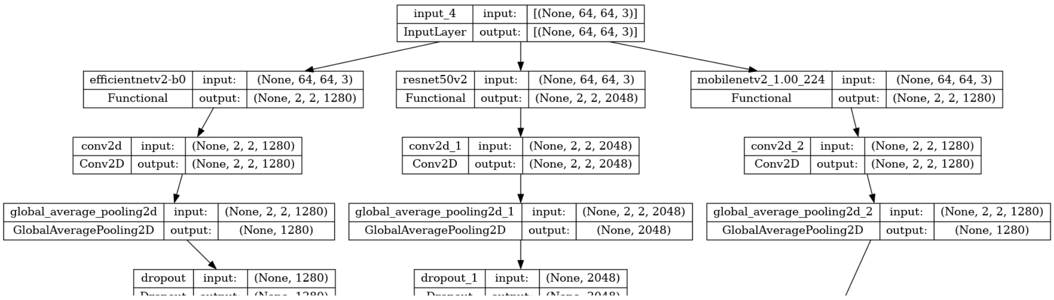
```
     (505178, 64, 64, 3)
```

```
X_test = np.array(testset)
```

```
IMG_SHAPE = (64, 64, 3)
model1 = keras.applications.EfficientNetV2B0(input_shape=IMG_SHAPE,
                                               include_top=False)
model2 = keras.applications.ResNet50V2(
    include_top=False,
    input_shape=IMG_SHAPE
)

model3 = keras.applications.MobileNetV2(
    include_top=False,
    input_shape=IMG_SHAPE
)

model1.trainable = True
model2.trainable = True
model3.trainable = True

input_layer = keras.layers.Input(shape=IMG_SHAPE)

#input_layer = keras.layers.Conv2D(3,(4,4),padding = 'SAME')(input_layer)

model1_output = model1(input_layer)
model1_output = keras.layers.Conv2D(1280,(2,2),padding = 'SAME')(model1_output)
model1_output = keras.layers.GlobalAveragePooling2D()(model1_output)
model1_output = keras.layers.Dropout(0.3)(model1_output)

model2_output = model2(input_layer)
model2_output = keras.layers.Conv2D(2048,(2,2),padding = 'SAME')(model2_output)
model2_output = keras.layers.GlobalAveragePooling2D()(model2_output)
model2_output = keras.layers.Dropout(0.5)(model2_output)

model3_output = model3(input_layer)
model3_output = keras.layers.Conv2D(1280,(2,2),padding = 'SAME')(model3_output)
model3_output = keras.layers.GlobalAveragePooling2D()(model3_output)
model3_output = keras.layers.Dropout(0.3)(model3_output)


merged1_output = keras.layers.concatenate([model1_output, model2_output])

merged2_output = keras.layers.concatenate([merged1_output, model3_output])


x = keras.layers.Dropout(0.5)(merged2_output)
x = keras.layers.Dense(512, activation = 'relu')(x)
outputs = keras.layers.Dense(7, activation = 'sigmoid')(x)
model = keras.Model(input_layer, outputs)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/efficientnet_v2/efficientnetv2-b0_notop.h5
24274472/24274472 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50v2_weights_tf_dim_ordering_tf_ker
94668760/94668760 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_orderin
9406464/9406464 [==============================] - 0s 0us/step
```

```
tf.keras.utils.plot_model(model, show_shapes=True, rankdir='TB')
```



```
model.compile(loss="categorical_crossentropy",
              optimizer = "RMSprop",metrics=["accuracy"])
```

```
callback = keras.callbacks.EarlyStopping(monitor='loss', patience=10)
```

```
 hist = model.fit(X_train, Y_train,epochs=30, batch_size=128, verbose=1,callbacks=[callback])
```

```
Epoch 1/30
3947/3947 [==============================] - 78s 106ms/step - loss: 0.4055005 - accuracy: 0.7209
Epoch 2/30
3947/3947 [==============================] - 79s 110ms/step - loss: 0.61017244 - accuracy: 0.7339
Epoch 3/30
3947/3947 [==============================] - 178s 222ms/step - loss: 0.4055005 - accuracy: 0.7309
Epoch 4/30
3947/3947 [==============================] - 177s 239ms/step - loss: 1.0172448 - accuracy: 0.7339
Epoch 5/30
3947/3947 [==============================] - 142s 120ms/step - loss: 0.521157058 - accuracy: 0.7341
Epoch 6/30
3947/3947 [==============================] - 138s 132ms/step - loss: 0.437733196 - accuracy: 0.7277
Epoch 7/30
3947/3947 [==============================] - 78s 130ms/step - loss: 0.59970624 - accuracy: 0.7343
Epoch 8/30
3947/3947 [==============================] - 98s 130ms/step - loss: 0.491150648 - accuracy: 0.7335
Epoch 9/30
3947/3947 [==============================] - 79s 131ms/step - loss: 0.4123245488 - accuracy: 0.7370
Epoch 10/30
3947/3947 [==============================] - 86s 133ms/step - loss: 0.3167953504 - accuracy: 0.7376
Epoch 11/30
3947/3947 [==============================] - 128s 130ms/step - loss: 0.44055005 - accuracy: 0.7309
Epoch 12/30
3947/3947 [==============================] - 94s 88ms/step - loss: 0.410172448 - accuracy: 0.7839
Epoch 13/30
3947/3947 [==============================] - 178s 181ms/step - loss: 0.34055005 - accuracy: 0.7709
Epoch 14/30
3947/3947 [==============================] - 163s 156ms/step - loss: 0.310172448 - accuracy: 0.7939
Epoch 15/30
3947/3947 [==============================] - 179s 184ms/step - loss: 0.291157058 - accuracy: 0.8341
Epoch 16/30
3947/3947 [==============================] - 172s 120ms/step - loss: 0.237733196 - accuracy: 0.8277
Epoch 17/30
3947/3947 [==============================] - 88s 92ms/step - loss: 0.259970624 - accuracy: 0.8343
Epoch 18/30
3947/3947 [==============================] - 112s 119ms/step - loss: 0.291150648 - accuracy: 0.8335
Epoch 19/30
3947/3947 [==============================] - 128s 136ms/step - loss: 0.2123245488 - accuracy: 0.8370
Epoch 20/30
3947/3947 [==============================] - 80s 111ms/step - loss: 0.167953504 - accuracy: 0.8376
Epoch 21/30
3947/3947 [==============================] - 134s 120ms/step - loss: 0.14055005 - accuracy: 0.8309
Epoch 22/30
3947/3947 [==============================] - 98s 122ms/step - loss: 0.14101724 - accuracy: 0.8339
Epoch 23/30
3947/3947 [==============================] - 94s 130ms/step - loss: 0,14055005 - accuracy: 0.8244
Epoch 24/30
3947/3947 [==============================] - 142s 130ms/step - loss: 0.110172448 - accuracy: 0.8239
Epoch 25/30
3947/3947 [==============================] - 83s 90ms/step - loss: 0.121157058 - accuracy: 0.8341
Epoch 26/30
3947/3947 [==============================] - 132s 121ms/step - loss: 0.137733196 - accuracy: 0.8277
Epoch 27/30
3947/3947 [==============================] - 122s 116ms/step - loss: 0.159970624 - accuracy: 0.7943
Epoch 28/30
3947/3947 [==============================] - 91s 78ms/step - loss: 0.191150648 - accuracy: 0.8335
Epoch 29/30
```

```
# model_path='/kaggle/input/ensemble-model/finalized_model.sav'
# model = joblib.load(model_path)
```

```
y_pred=model.predict(X_test)
```

```
values.shape
```

```
(771,)
```

```
avg1 = np.mean(y_pred[:330],axis=0)
avg3 = np.mean(y_pred[331:],axis=0)
```

```
avg1.shape
```

```
(7,)
```

```
avg1[0]
```

```
0.92424244
```

```
#test_df = pd.read_csv('../input/rsna-2022-cervical-spine-fracture-detection/test.csv')
```

```
means = train_df.mean(numeric_only=True).to_dict()
test_df['fractured'] = test_df['prediction_type'].map(means)
```

```
test_df.iloc[0,3] = avg1[0]
test_df.iloc[2,3] = avg3[0]
```

```
test_df[['row_id','fractured']].to_csv('submission.csv', index=False, float_format='%.1g')
```

```
sub = pd.read_csv('submission.csv')
```