Sardar Patel Institute of Technology,Mumbai

Department of Computer Science Engineering

B.E. Sem-VII- PE-IV (2024-2025)

IT 24 - AI in Healthcare

Experiment 8: Implement KERAS or Tensorflow model for automatic detection of abnormalities

Name: Adwait Purao UID : 2021300101 Date : 6/11/24

Objective:

To understand the basics of neural networks ,Convolutional Neural Network (CNN) or other deep learning models and implement them using Keras and TensorFlow,

Theory:

Overview of abnormalities in healthcare,CNN and other deep learning models, Keras and TensorFlow libraries.

Experiment:

1.Install Keras and TensorFlow

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
import os
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers import Flatten,Dense,Dropout,BatchNormalization
from tensorflow.keras.models import Model,Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import GlobalAveragePooling2D
```

```
  Using cached absl_py-2.1.0-py3-none-any.whl (133 kB)
Collecting keras>=3.0.0
  Downloading keras-3.3.3-py3-none-any.whl (1.1 MB)
Requirement already satisfied: setuptools in c:\users\aspur\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tenso
rflow) (61.2.0)
Collecting opt-einsum>=2.3.2
  Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Collecting numpy<2.0.0,>=1.23.5
  Downloading numpy-1.26.4-cp39-cp39-win_amd64.whl (15.8 MB)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\aspur\anaconda3\lib\site-packages (from tensorflow-intel==2.1
6.1->tensorflow) (1.42.0)
Collecting libclang>=13.0.0
  Using cached libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
Collecting ml-dtypes~=0.3.1
  Downloading ml_dtypes-0.3.2-cp39-cp39-win_amd64.whl (127 kB)
Collecting google-pasta>=0.1.1
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Collecting h5py>=3.10.0
  Downloading h5py-3.11.0-cp39-cp39-win_amd64.whl (3.0 MB)
```

2. Dataset Preparation

a.Loading the dataset

In [3]:
```
batch_size = 32
img_height = 150
img_width = 150
```

In [4]:
```
data_dir='ct-ori-dataset'
train=tf.keras.utils.image_dataset_from_directory(data_dir,image_size=(img_height,img_width),
                                                  validation_split=0.1,
                                                  subset='training',seed=123)
val=tf.keras.utils.image_dataset_from_directory(data_dir,image_size=(img_height,img_width),
                                                  validation_split=0.2,
                                                  subset='validation',seed=123)
```

```
Found 12446 files belonging to 4 classes.
Using 11202 files for training.
Found 12446 files belonging to 4 classes.
Using 2489 files for validation.
```

In [5]:
```
label_to_class_name = dict(zip(range(len(train.class_names)), train.class_names))
label_to_class_name
```

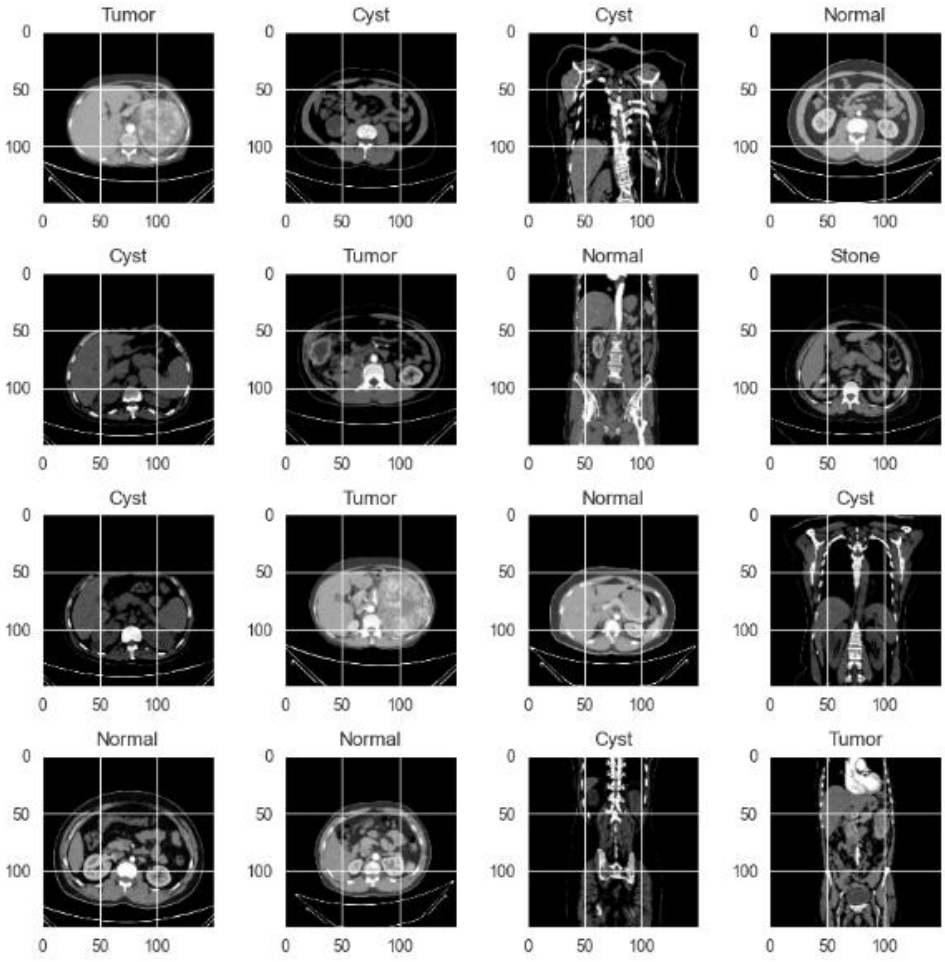
Out[5]:
```
{0: 'Cyst', 1: 'Normal', 2: 'Stone', 3: 'Tumor'}
```

In [6]:
```
for image_batch, labels_batch in train:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
(32, 150, 150, 3)
(32,)
```

b.Preprocessing the data

In [7]:
```
data_iterator = train.as_numpy_iterator()
batch = data_iterator.next()
fig, ax = plt.subplots(nrows=4, ncols=4, figsize=(10, 10))
for i in range(4):
    for j in range(4):
        index = i * 4 + j
        ax[i, j].imshow(batch[0][index].astype(int))
        ax[i, j].set_title(label_to_class_name[batch[1][index]])
plt.subplots_adjust(wspace=0.4, hspace=0.4)
plt.show()
```



3.Model Building

In [8]:
```
train=train.map(lambda x,y:(x/255,y))
val=val.map(lambda x,y:(x/255,y))
```

In [9]:
```
AUTOTUNE = tf.data.AUTOTUNE

train = train.cache().prefetch(buffer_size=AUTOTUNE)
val = val.cache().prefetch(buffer_size=AUTOTUNE)
```

4. Model Training

Mobile-Net Training

In [10]:
```
mobile_net = Sequential()

pretrained_model= tf.keras.applications.MobileNetV2(include_top=False,
                   input_shape=(150,150,3),
                   pooling='max',classes=4,
                   weights='imagenet')


mobile_net.add(pretrained_model)
mobile_net.add(Flatten())
mobile_net.add(Dense(512, activation='relu'))
mobile_net.add(BatchNormalization())  # Batch Normalization Layer
mobile_net.add(Dropout(0.5))

mobile_net.add(Dense(4, activation='softmax'))
pretrained_model.trainable=False
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input sh
ape (224, 224) will be loaded as the default.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ord
ering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 [==============================] - 39s 4us/step
```

In [13]:
```
epochs=15
history = mobile_net.fit(
  train,
  validation_data=val,
  epochs=epochs
)
```

```
Epoch 1/15
351/351 [==============================] - 114s 310ms/step - loss: 0.4080 - accuracy: 0.8564 - val_loss: 0.1676 - val_accuracy:
0.9373
Epoch 2/15
351/351 [==============================] - 88s 252ms/step - loss: 0.1292 - accuracy: 0.9557 - val_loss: 0.0594 - val_accuracy:
0.9847
Epoch 3/15
351/351 [==============================] - 87s 249ms/step - loss: 0.0835 - accuracy: 0.9741 - val_loss: 0.0578 - val_accuracy:
0.9839
Epoch 4/15
351/351 [==============================] - 101s 287ms/step - loss: 0.0632 - accuracy: 0.9794 - val_loss: 0.0632 - val_accuracy:
0.9779
Epoch 5/15
351/351 [==============================] - 87s 248ms/step - loss: 0.0540 - accuracy: 0.9827 - val_loss: 0.0258 - val_accuracy:
0.9932
Epoch 6/15
351/351 [==============================] - 88s 250ms/step - loss: 0.0448 - accuracy: 0.9847 - val_loss: 0.0435 - val_accuracy:
0.9843
Epoch 7/15
351/351 [==============================] - 90s 257ms/step - loss: 0.0381 - accuracy: 0.9869 - val_loss: 0.0307 - val_accuracy:
0.9916
Epoch 8/15
351/351 [==============================] - 85s 242ms/step - loss: 0.0387 - accuracy: 0.9869 - val_loss: 0.0458 - val_accuracy:
0.9871
Epoch 9/15
351/351 [==============================] - 84s 239ms/step - loss: 0.0374 - accuracy: 0.9874 - val_loss: 0.0379 - val_accuracy:
0.9875
Epoch 10/15
351/351 [==============================] - 87s 249ms/step - loss: 0.0352 - accuracy: 0.9884 - val_loss: 0.1365 - val_accuracy:
0.9574
Epoch 11/15
351/351 [==============================] - 87s 249ms/step - loss: 0.0326 - accuracy: 0.9893 - val_loss: 0.0385 - val_accuracy:
0.9859
Epoch 12/15
351/351 [==============================] - 87s 249ms/step - loss: 0.0259 - accuracy: 0.9919 - val_loss: 0.0237 - val_accuracy:
0.9912
Epoch 13/15
351/351 [==============================] - 87s 249ms/step - loss: 0.0247 - accuracy: 0.9921 - val_loss: 0.0235 - val_accuracy:
0.9920
Epoch 14/15
351/351 [==============================] - 88s 250ms/step - loss: 0.0252 - accuracy: 0.9913 - val_loss: 0.0263 - val_accuracy:
0.9912
Epoch 15/15
351/351 [==============================] - 86s 245ms/step - loss: 0.0272 - accuracy: 0.9911 - val_loss: 0.0310 - val_accuracy:
0.9908
```

5. Model Evaluation

In [14]:
```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss)+1)
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [15]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

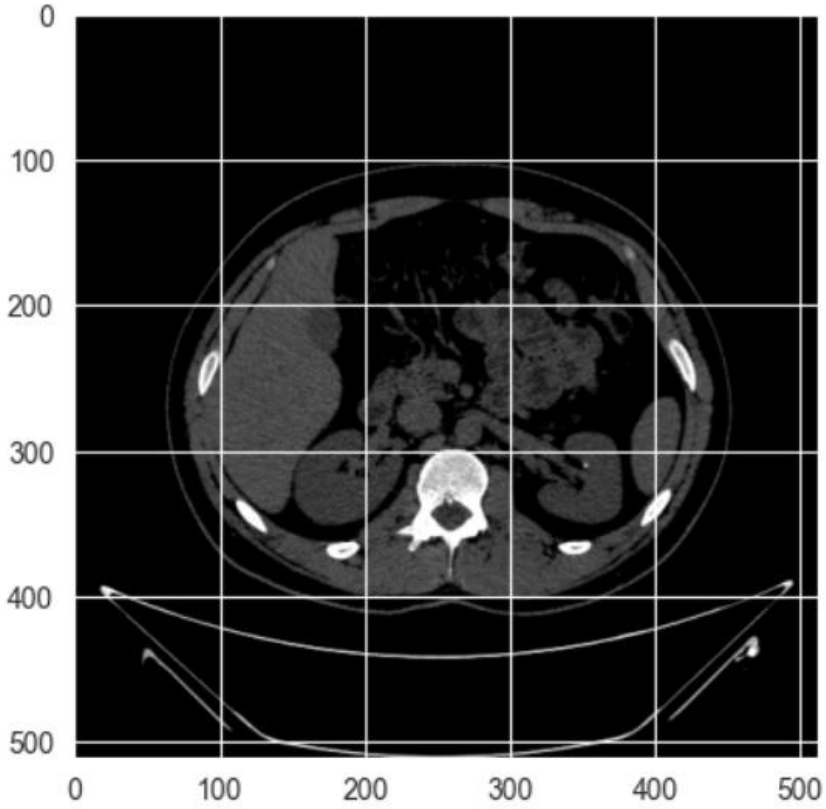
In [ ]:
```
interp.print_classification_report()
```

	precision	recall	f1-score	support
Kidney_stone	0.98	0.96	0.97	165
Normal	0.96	0.98	0.97	181
accuracy			0.97	346
macro avg	0.97	0.97	0.97	346
weighted avg	0.97	0.97	0.97	346

Using performance metrics accuracy, precision, recall, and AUC-ROC.

6.Testing on new data

```
In [21]: img = cv2.imread('test.jpg')
         plt.imshow(img)
         plt.show()
         resize = tf.image.resize(img, (150,150))
         yhat = loaded_model.predict(np.expand_dims(resize/255, 0))
         max_index = np.argmax(yhat)
         label_to_class_name[max_index]
```



Conclusion:

The study illustrates that Convolutional Neural Networks (CNNs), utilizing Keras and TensorFlow, serve as powerful tools for identifying abnormalities in healthcare data. This approach exhibits high levels of accuracy and efficiency in processing medical images and other health-related information, showcasing its potential for practical use in early and accurate diagnosis.